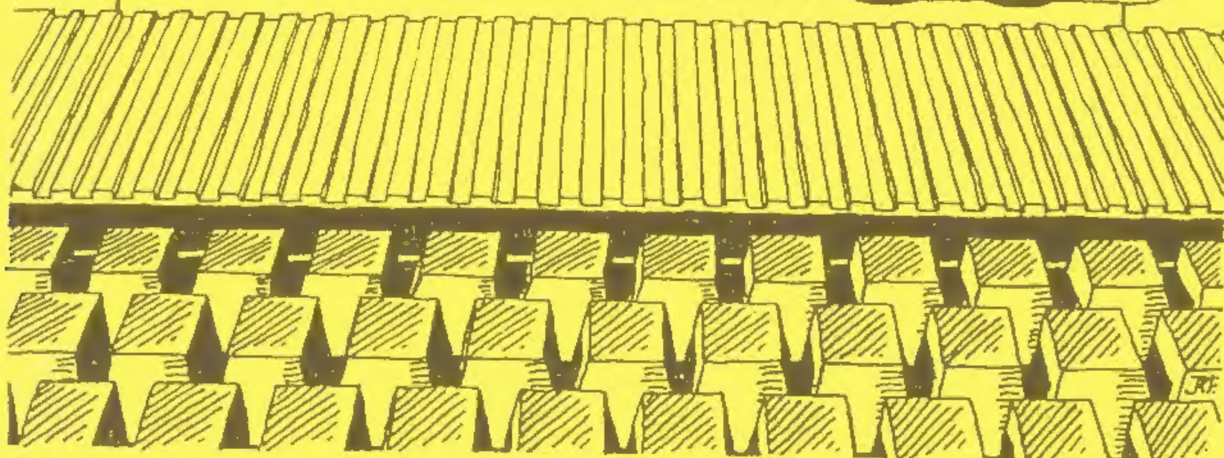
ATOM NIEUWS

JAARGANG
NUMMER



Bestuur:

Voorzitter:	Secretaris:	Penningmeester:
N.Stad	J.Hartog	T.Rutten
Plataanweg 47	Keyenbergseweg 60	Berkenlaan 24
1544 PB Zaandijk	6871 WK Renkum	3737 RN Groenekan
Tel.075-280808	Tel.08373-13757	Tel.03461-3495

Clubwinkel:	Hardware cie.:	Redactie Atom Nieuws:
P.Grevelt	Y.Tuk	H. de Ruiter
Emmastraat 22	Postbus 257	Polarisstraat 25
1782 PD Den Helder	2980 AG Ridderkerk	8303 AC Emmeloord
Tel.02230-23453		

Contributie 1987: fl. 60.00
Leden buitenland: fl. 75.00

Rekeningnummers Atom Computer Club:
Giro 5244293 Bank 52.84.69.010

Redactie Atom Nieuws:
Joop Ballijns
Harry de Ruiter
Gerhard Visser

Ledenadministratie:	Datasheets:
S.van Leeuwen	G.Akkerman
Kompasstraat 32	Wikke 1
1973 PX IJmuiden	1273 BR Huizen
Tel.02550-22435	Tel.02152-60294

Uiterste datum inlevering copy:
nr. 87-1 01-02-1987

De clubwinkel:

80-koloms videokaart, incl.alle onderdelen behalve de RAM-IC's, bouwpakket	fl.130.00
80-koloms videokaart,gebouwd en getest,excl.RAM-IC's	fl.180.00
Geheugenkaart; 16 kByte,excl.onderdelen	fl. 40.00
Schakelkaart; meerdere EPROM's op Axxx	fl. 47.50
Minischakelkaart	fl. 16.00
Herdruk ACORN NIEUWS 1982; 97 pag.wetenswaardigheden	fl. 6.00
ATOM NIEUWS jaargang 1983, ruim 450 pag.	fl. 30.00
ATOM NIEUWS jaargang 1984, ruim 650 pag.	fl. 35.00
ATOM NIEUWS jaargang 1985, ruim 650 pag.	fl. 35.00
ATOM-WARE 1; Basic interpreter van de ATOM, 98 pag.	fl. 6.00
ATOM-WARE 2; ATOM Disc Operating System,68 pag.	fl. 5.00
ATOM-WARE 3; ATOM Monitor Operating System,80 pag.	fl. 5.00

Levering;
Bij voorkeur via uw regionale penningmeester, eventueel rechtstreeks bij de penningmeester van de federatie.Bij rechtstreekse bestelling dient u het bedrag van het gewenste artikel te storten op de giro van de federatie onder vermelding van de naam van het artikel en uw lidnummer.Uw betaling dient vermeerderd te zijn met fl. 4.00 voor portokosten.

ATOM NIEUWS is een uigave van de federatie ATOM computerclubs Ned/Belgie en verschijnt 6 - 8 keer per jaar.
De redactie gaat er vanuit dat de ingezonden copy gemaakt is door de inzender tenzij in de publicatie uitdrukkelijk anders is vermeld.De aansprakelijkheid betreffend de auteursrechten ligt dan ook volledig bij de inzender.

NAGEKOMEN:

Wist u dat in CANADA ook een ATOM-club actief is? (ong.100 leden). Wist u, dat de informatie uitwisseling inmiddels al op gang gekomen is? Weet u dat de redactie probeert wereldwijd contact te krijgen met andere ATOM clubs? Kent u ze? Laat het ons weten!!!!!!

DDCOPY	8200	C2B2	00816	04C		#SETCAT	2900	CE86	0033F	046
DCOPY	8200	C2B2	00AAE	055		#PCAT2	2900	CE86	0047F	04A
BACKUP	8200	C2B2	012E3	060		#PCAT1	2900	CE86	0035F	04F
GP50DMP	2900	C2B2	0063F	073		#PC-SOUR	2900	AFAF	002FC	053
AUTOSHW	2900	C2B2	003BF	07A		#PC-SORT	2800	2800	0007F	056
GROLSCH	8000	8000	01800	07E		#DISMENU	9000	CE86	0022F	057
CAT	8000	8000	01800	096		#DFSOUR	2900	AFAF	007EA	05A
ROLLS	8000	8000	01800	0AE		#DFCAT	2700	CE86	0016F	062
LOVEBOA	8000	8000	01800	0C6		#DISKCAT	2900	CE86	0067F	064
EYE	8000	8000	01800	0DE	C	CX-9345	1000	1D00	01000	06B
MELINA	8000	8000	01800	0F6	C	DBOX2.1	A000	A000	01000	07B
SW	2900	AFAF	00571	002	C	CX19345	0900	1D00	01700	08B
E-PROM	2900	AFAF	0046F	008	C	CX-8040	6800	1D00	01800	0A2
GRAFIEK	2900	AFAF	00070	00D	C	CX-2.8h	A000	A000	01000	0BA
JOYTEK	2900	AFAF	00DF5	00E	C	UITLEG	2800	A071	00469	0CA
TEKBEST	8200	8200	01300	01C	S	SURFscr	8000	8000	017FF	0C9
KAARTEN	2900	AFAF	00A2F	02F	S	SURF	2900	CE86	0146B	0E1
KASSA 2	2900	AFAF	005B4	03A	A	SORT	0400	0400	00072	002
LEEFT	2900	AFAF	000B8	040		VOORRD	2900	AFAF	01A76	003
LEUK	2900	AFAF	0007F	041	X	voortxt	2800	A071	00C13	01E
MARCEL	2900	AFAF	00413	042		MASTER	0000	0000	00500	02B
OVERDWA	2900	AFAF	005DD	047	G	GMSNW.M	5900	CE86	0244A	034
PRINSES	2900	AFAF	00FD9	04D	G	GMSNW.A	2900	CE86	02B2D	059
RAMTEST	2900	AFAF	001C9	05D	G	GMSNrun	2900	CE86	000CD	085
WASMACH	2900	AFAF	00B9F	05F	G	GMSNW.B	9800	9800	00800	086
CALLIGR	2900	2900	00F63	002	G	#SCREEN	8000	8000	01800	08E

	#1	2900	CE86	000CF	002
	#FCAT	3000	3000	037FF	003
	#startdc	2900	CE86	001BF	03B
I	KURUVU	2900	2900	009B3	002
I	PANFAST	2900	CE86	00514	00C
D	DUOTASK	5000	5000	00994	012

Zie pag 3 onderaan!!!!!

VAN DE VOORZITTER

Hier dan weer een artikel van mijn(onze) hand(en).
Allereerst moet mij even iets van het hart.
Ik heb van een heleboel mensen steeds boze reacties gehad over het late uitkomen van de 80-kolomskaart, maar nu hij er is wordt hij maar bar weinig besteld. Dit is jammer want het is zo'n schitterende kaart voor een minimum aan kosten. Dus mensen, laten we allemaal van onze ATOM een computer met een prachtig beeldscherm maken. Er komen dan ook meer interessante programma's los, zoals bijvoorbeeld een 80-koloms tekstverwerker. Overigens is het programma ED-80 uit het vorige ATOM NIEUWS wat op de regio schijf stond, bedoeld voor de elektuur 80 kolomskaart maar zal snel worden aangepast voor onze eigen kaart. Ook voor een spreadsheet (calc) is onze 80 kolomskaart te gebruiken. Inplaats van de huidige 4 kolommen van 8 tekens heb je dan 10 kolommen van 8 tekens breed op 1 beeldscherm.

Dan nu een ander punt namelijk de G-DOS kaart.
Verderop in dit nummer staat ook nog een artikel over dit onderwerp, maar ik wou nu iets vanuit het bestuur kwijt. De GDOS kaart is tijdelijk opgehouden omdat deze zomer een andere parallel lopende kaart werd aangeboden die meer mogelijkheden in zich bergt. Peter Huisken heeft in alle stilte een Z-80 (second processor) kaart ontwikkeld, gebouwd en getekend. Tevens heeft hij hiervoor een compleet besturingssysteem geschreven, welke gelijke mogelijkheden heeft als het CP/M besturingssysteem. Voor CP/M is ongelooflijk veel software (Public Domain en Profesioneel) verkrijgbaar waardoor onze ATOM nog meer in achting stijgt voor wat de mogelijkheden betreft. Op 13 September tijdends de ALV is hier uitgebreid over gediscusieerd en er is besloten om in principe beide kaarten door te zetten op basis van voorbestellingen. Dit is te regelen door 80 gulden over te maken op de giro rekening van de federatie onder vermelding van GDOS of Z-80 kaart, en vergeet niet je lidmaatschapnummer te vermelden. Voor verdere informatie verwijs ik naar het artikel verderop in ATOM NIEUWS.

Verder kan ik melden dat de contributie voor 1987 op 60 gulden is gesteld. Het kan zijn dat er in dit nummer nog niet de acceptgirokaart is ingesloten. Je kunt wel alvast betalen naar de girorekening van de federatie onder vermelding van je lidmaatschapnummer en contributie 1987.

groeten,
Nico Stad

MODEM SOFTWARE (OFTEWEL HOE VERSTUUR IK MIJN PROGRAMMA TERWIJL IK KOFFIE DRINK).

Bij gebruik van het atom modem met de save/load methode komt het (bij mij in ider geval) regelmatig voor dat de communicatie wordt afgebroken met een sum-error. Bij veel storing op de telefoon lijn kan het oversturen zo een tijdrovende (en irritante) zaak worden. Met het programma MODEM hoop ik deze problemen op te lossen.

Het programma moet zowel aan de zend- als aan de ontvang zijde aanwezig zijn, zodat er tweeweg communicatie mogenlijk is. Na ieder verstuurt blok van 256 bytes (1 page) geeft de ontvanger een melding of het blok goed is overgekomen. Is het blok niet goed overgekomen, dan wordt dit nogmaals gestuurd.

Het oversturen wordt gestart met een link naar het begin van het programma, waarna het prg. om S(end), R(eceive) of E(nd) in te toetsen. Bij S wordt er gevraagd om de eerste en de laatste te versturen page, bij R wordt alleen de start page gevraagd. Dit moet worden ingegeven als 2 tekens hexadecimaal. Door als pagenr. 00 in te geven wordt teruggesprongen naar de voorafgaande vraag. Na het ingeven van het pagenr. gaat de ontvanger staan wachten op het eerste blok data. De zender komt met de opmerking "hit return". Als nu op de returntoets wordt gedrukt, wordt met oversturen begonnen. Van ieder overgezonden blok data wordt melding gemaakt op het scherm. De punt die achter het pagenr. verschijnt geeft bij zenden aan, dat met zenden in begonnen, terwijl dit bij ontvangen betekent dat het blok binnen is. De "sum", die zich aan het einde van ieder blok bevindt, bestaat uit een heuse CRC, zoals ook gebruikt wordt voor het testen van de diverse eproms. Als antwoord wordt door de ontvanger (na het controleren van de crc) een ack of nack gestuurd. Dit wordt ook op het scherm zichtbaar als text "ack" of "nack".

Het verstuurde blok is als volgt opgebouwd:

STX	geeft start van het blok aan. (ontvanger wacht hierop).
DATA	blok heeft lengte tussen 265 en 512 bytes. De gebruikte controle tekens worden nl. voorafgegaan door een dle. Deze tekens zijn: stx, etb, eot, dle.
ETB of EOT	etb geeft het einde van het blok aan. eot geeft het einde van het blok en van de communicatie aan. (dus alleen het laatste blok wordt afgesloten met eot).
2 BYTES CRC	de checksum over het blok data, inclusief de extra dle's en de etb of eot. Niet over de stx.

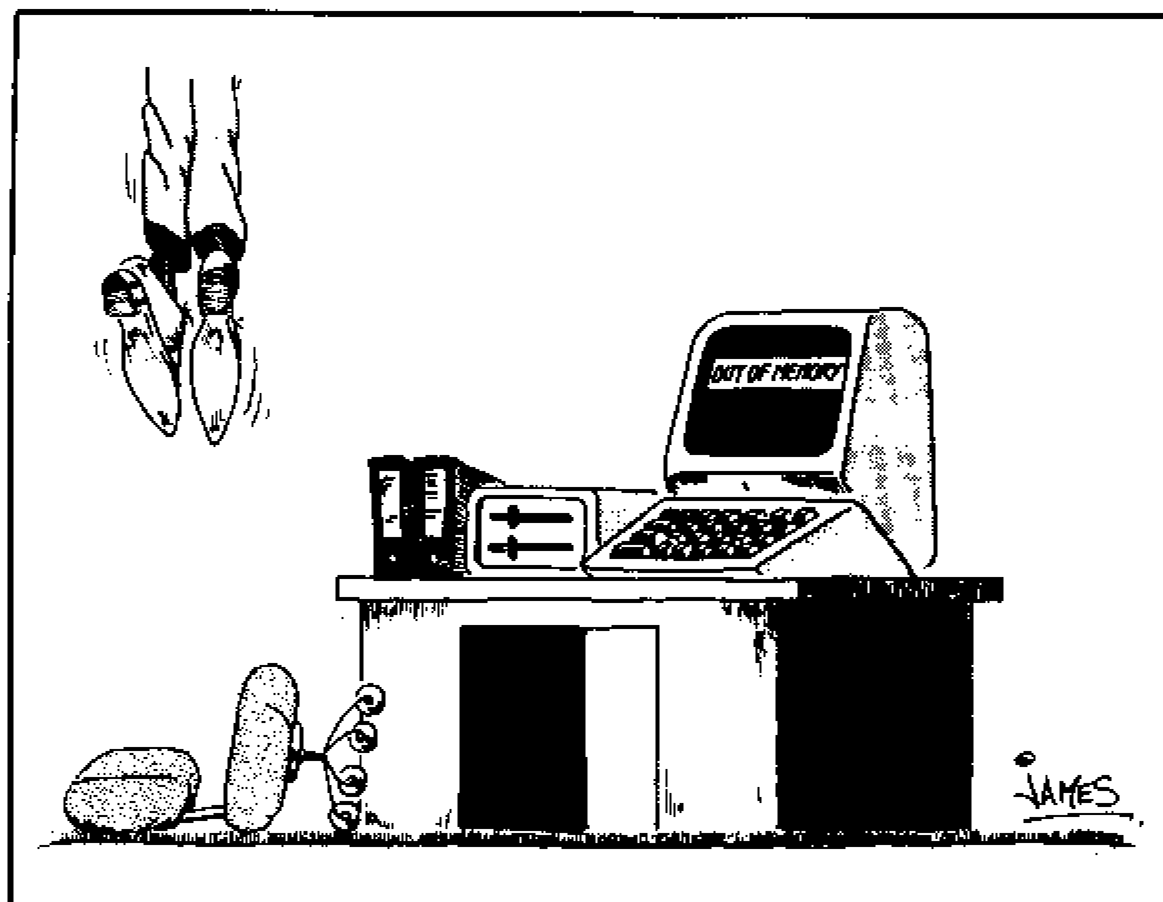
De genen die met "echte" communicatie te maken hebben, zullen hebben opgemerkt dat het protocol min of meer van de transparante uitvoering het bsc (bisynchronious comm.) protocol is afgekeken. Ik heb me alleen niet bezig gehouden met "timers" en "counters", wat kan betekenen dat bij een slechte lijn de computers rustig een half uurtje hetzelfde blok data kunnen uitwisselen, waarbij het beter is om met BREAK het programma af te breken en het op een later tijdstip nog eens te proberen.

Voor het werkelijk versturen is gebruik gemaakt van de bget en bput routines, welke direct worden aangeroepen (dus niet via #214 en #216) zodat er niet teruggeschakeld hoeft te worden bij gebruik van diskette (dos). Dit heeft tot gevolg dat er altijd op 300 baud wordt gestuurd. De gebruikte adressen zijn voor bget #FBEE en bput #FC7C.

Mocht iemand met 1200 baud willen werken, dan moeten deze adressen in het programma worden aangepast voor de 1200 baud routines. Voor P-Charme zijn dit : bget #ADE0 en bput #AE0E.

Voor vragen en opmerkingen hou ik mij aanbevolen.

Dick Bronsdijk.



Meer geld informatica ondanks bezuinigingen

Scholen krijgen meer apparatuur


Minister W. J. Deetman

Het Rijk stelt de komende vier jaar honderden miljoenen guldens extra ter beschikking voor de verbetering van (computer)apparatuur en inventarissen op scholen. Dit bedrag loopt zelfs op tot 300 miljoen gulden in 1990. Doel van deze financiële injectie is in de eerste plaats de relatie onderwijs-arbeid te versterken. Dit houdt in dat het hoger- en middelbaar beroepsonderwijs meer computerapparatuur ter beschikking krijgt. Ook wordt meer aandacht geschonken aan de ontwikkeling van programma's en de nascholing van leerkrachten op het gebied van informatica.

Dit staat in de memorie van toelichting op de begroting Onderwijs en Wetenschappen. De 300 miljoen gulden extra (in 1990 bereikt) is des te opmerkelijker omdat Onderwijs daarnaast ook in 1990 circa 1 miljard gulden moet hebben bezuinigd.

Dit gaat echter niet ten kostte van de informatica-ontwikkeling. Het voornemen bestaat om in de derde en in de vierde klassen van havo en vwo een lesuur informatica als afzonderlijk vak verplicht te stellen.

Vaste voet

Minister Deetman schrijft in zijn begroting: "De nieuwe informatietechnologie heeft vaste voet gekregen in de verschillende sectoren van het onderwijs. Algemeen is het besef gegroeid dat deze technologische vernieuwing een belangrijk aanknopingspunt is voor de bevordering van economische groei."
De overheid wil (overeenkomstig het standpunt van de commissie Pannenborg over het Surf-plan; de samenwerkende universitaire rekencentra) voorrang geven aan het opzetten van een landelijk datanetwerk en aan voorzieningen voor de hogescholen voor beroepsonderwijs.

Micro-elektronica

Bovendien zal het onderzoek in de micro-elektronica worden gestimuleerd.
Het middelbaar beroepsonderwijs wordt de komende jaren ingrijpend gewijzigd in sectorscholen. Dit levert een bezuiniging op van 150 miljoen gulden per jaar. Aan de andere kant kunnen MBO-scholen aanmerkelijk flexibeler werken door een globaler bekostiging en een sterke bevordering van de autonomie op een wijze die doet denken aan de operatie schaalvergroting, taakverdeling en concentratie voor het HBO.
Het MBO-nieuwe stijl wordt in vier sectoren verdeeld: een technische, een economisch-administratieve, een dienstverlenende en gezondheids- en een agrarische sector. In opdracht van Onderwijs is ook een eerste studie naar de mogelijkheden van de nieuwe media zoals beeldplaat en viewdata verricht door het European Institute of Education and Social Policy te Parijs. Aan het einde van deze regeerperiode zal meer duidelijkheid zijn verkregen over de wenselijkheid van bepaalde educatieve toepassingen.

Luier zorgt voor gladde floppy

Extra goed absorberende luiers hebben geleid tot extra gladde floppies, zo meldt het Amerikaanse blad Business Week. Om de concurrent Pampers bij te blijven ontwikkelde het Japanse bedrijf Kao luiers op basis van minuscule plastic kraaltjes die aanmerkelijk beter bleken te absorberen dan papier. Met de inmiddels opgedane materiaalkennis ontwikkelde Kao vervolgens een flexibel schijfje met een superglad oppervlak.
Waarschijnlijk om ongewenste associaties te voorkomen zullen er geen Kao floppies op de markt verschijnen. Men zal alleen de materialen voor de vervaardiging van de supergladde schijven leveren.

ATV: Automatisch Tijd Verdrijf

Zie pag 3 onderaan!!!!!

Ik heb gehoord dat sommigen van ons het leuk zouden vinden om een soort voor stukje te hebben dat direct op het beeld schrijft en laat zien van wie het is.
Alleen het probleem was hoe doen we dit.
Eerst moet er een ontwerpje gemaakt worden.
Je moet er rekening mee houden dat het in clear0 gezet wordt.
Hier onder volgt een voorbeeld:

```
 10 P.#12;CLEAR0
 20 P."        ( C ) ACORN CLUB
 30 MOVE20,30
 40 DRAW20,40
 50 DRAW25,30
 60 DRAW25,40
 70 MOVE30,30
 80 DRAW30,40
 90 MOVE35,30
100 DRAW35,40
110 MOVE30,35
120 DRAW35,35
130 P.'''''''
140 P."  * P R O D U C T I O N S *"'
150 P.'"   DIT IS EEN AANKONDIGING"'
160 P.'"     VOOR HET KOMENDE"'
170 P.'"  PROGRAMMA VAN ERROR 94"
180 END
```

Wanneer je dit runt dan zie je een ontwerp voor Noord-Holland.
De bedoeling is alleen om er een beeld loading van te maken.
De regel 180 wordt hier verwijderd en het volgende bij gevoegd:

```
180 *NOMON
190 FCOS
200 *SA." naam "8000 8200
210 P.#7#7#7
220 END
```

Na het runnen en het opnemen de band terug spoelen.
Nu kun je het voorprogrammaatje gewoon laden en dan zie je dat het direct op het beeld komt.
Wil je nu je eigen programma er aan vastkoppelen dan scrijf je op #8200 *RUN" naam eigen programma " en verander dan het adres op regel 130 in de top.
Veel plezier er mee.

Ron Siekman

Dit programma is gemaakt omdat er aan mij vaak gevraagd werd omeens een programma te schrijven dat het maken van kleine tekeningen vergemakkelijkt.
Nou hier is het dan.
De toetsen die je moet gebruikenwordt door het programma zelf verteld.
Na dit programma volgt nog een voorbeeld met een kleine uitleg.Veel plezier bij het intikken.

```
 10 REM (C) ACORN CLUB
 20 DIM L(63),A(20)
 30 !L=#6E3E4477;L!4=#46796B4D;L!8=#4E1B4F7F;L!12=#2233683F
 40 L!16=#223B6C38;L!20=#6A33021B;L!24=#22664C19
 50 L!28=#4A1D0276;L!32=#46172031;L!36=#66336457
 60 L!40=#2E4B0E1B;L!44=#6A2B2E5B;L!48=#64310246
 70 L!52=#64716449;L!56=#0C492C5E;L!60=#2E3A
 80 CLEAR 4
 90 FOR I=10 TO 130 STEP15
100    MOVEI,10
110    DRAWI,145
120 NEXT I
130 FOR I=10 TO 145 STEP15
140    MOVE10,I
150    DRAW130,I
160 NEXT I
170 $A="FIGURE DESIGN RF SIEKMAN";MOVE22,170;GOSUB a
180 $A="128 64 32 16  8  4  2  0";MOVE11,137;GOSUB a
190 $A="A UP";MOVE140,135;GOSUB a
200 $A="Z DOWN";MOVE140,125;GOSUB a
210 $A="N LEFT";MOVE190,135;GOSUB a
220 $A="M RIGHT";MOVE190,125;GOSUB a
230 $A="D FILL";MOVE140,105;GOSUB a
240 $A="C CLEAR";MOVE190,105;GOSUB a
250 $A="RESULT";MOVE165,70;GOSUB a
260 FOR D=#8000 TO #9800
270    ?D=?D:#FFFFFFFF
280 NEXT D
290 X=16;Y=16;P=175;Q=40
300 PLOT14,X,Y;PLOT14,P,Q
310 KEY Z
320  IF Z=CH"N" THEN X=X-15;PLOT14,X,Y;PLOT14,(X+15),Y
330  IF Z=CH"N" THEN P=P-2;PLOT14,P,Q;PLOT14,(P+2),Q
340  IF Z=CH"M" THEN X=X+15;PLOT14,X,Y;PLOT14,(X-15),Y
350  IF Z=CH"M" THEN P=P+2;PLOT14,P,Q;PLOT14,(P-2),Q
360  IF Z=CH"A" THEN Y=Y+15;PLOT14,X,Y;PLOT14,X,(Y-15)
370  IF Z=CH"A" THEN Q=Q+2;PLOT14,P,Q;PLOT14,P,(Q-2)
380  IF Z=CH"Z" THEN Y=Y-15;PLOT14,X,Y;PLOT14,X,(Y+15)
390  IF Z=CH"Z" THEN Q=Q-2;PLOT14,P,Q;PLOT14,P,(Q-2)
400  IF Z=CH"D" THEN GOSUB b
410  IF Z=CH"C" THEN GOTO 20
420 WAIT;WAIT;WAIT;GOTO 310
430aFOR I=0 TO LEN(A)-1
440    C=A?I;GOSUB c
450 NEXT I
```

```
460cIF C=32 THEN PLOT0,6,0;RETURN
470 IF C<48 THEN RETURN
480 C=C-48
490 IF C<10 THEN GOSUB d;PLOT0,2,0;RETURN
500 C=C-17
510 IF C<0 OR C>25 THEN RETURN
520 C=2*C+10
530 GOSUB e
540 PLOT0,3,0
550 RETURN
560eD=C
570 GOSUB d
580 C=D+1
590 GOSUB d
600 RETURN
610dC=L?C;PLOT(C&1),0,2
620 C=C/2;PLOT(C&1),2,0
630 C=C/2;PLOT(C&1),0,-2
640 C=C/2;PLOT(C&1),-2,0
650 C=C/2;PLOT(C&1),0,-2
660 C=C/2;PLOT(C&1),2,0
670 C=C/2;PLOT(C&1),0,2
680 RETURN
690bMOVE(X-5),(Y-4)
700 FOR A=X-5 TO X+8
710     B=Y+8
720     PLOT6,A,B
730     MOVE A,(Y-5)
740 NEXT A
750 MOVE(X+8),(Y-5)
760 PLOT6,(X+8),(Y+1)
770 MOVE(X-5),(Y-4)
780 PLOT6,(X-5),(Y+1)
790 MOVEP,Q
800 FOR V=P-1 TO P
810     W=Q+1
820     PLOT6,V,W
830     MOVEV,Q
840 NEXT V
850 RETURN
```

Na dat je dit programma hebt gerund zie je dat doormiddel vande toets D de vlakken kunt in kleuren en ook weer uitvegen.
Nu volgt de verdere uitwerking. Nadat je een tekening gemaakt hebt worden de getallen als volgt opgeteld:

bij de eerste regel heb je ingekleurd
de 16 en 8
2e 32 en 4
3e 32 en 4
4e 32,16,8 en 4
5e 6e 7e 8e 32 en 4

Dit ziet er als een A uit.

Nu tellen we de getallen op.
b.v. bij de 4e regel word dit :

32+16+8+4=60

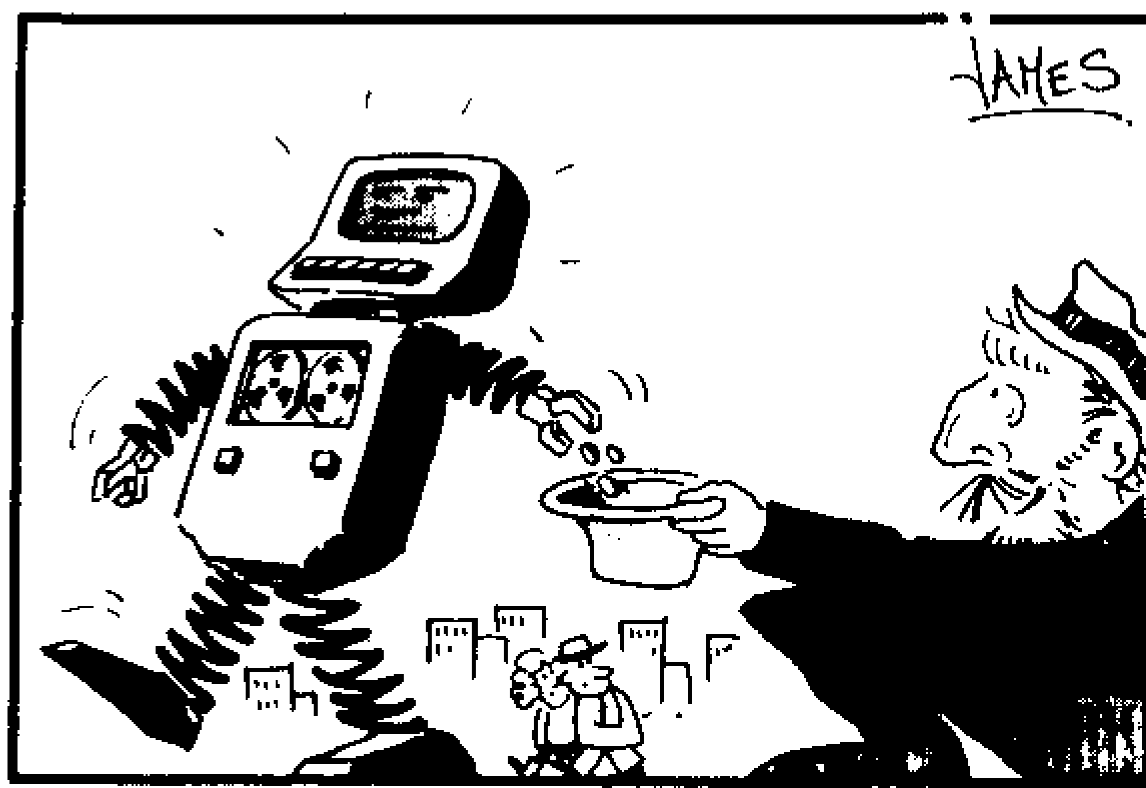
Dit doen we met alle getallen. Nu gaan we het gebruiken en proberen om het figuurtje op het scherm te krijgen.

```
 10 REM plaats bepaling
 20 Z=#90D0
 30 REM mode waar in het getekend moet worden
 40 CLEAR4
 50 REM ontwerp tekenening
 60 ?Z=24;Z=Z+32;?Z=36;Z=Z+32
 70 ?Z=36;Z=Z+32;?Z=60;Z=Z+32
 80 ?Z=36;Z=Z+32;?Z=36;Z=Z+32
 90 ?Z=36;Z=Z+32;?Z=36;Z=Z+32
100 E.
```

Ik hoop dat jullie er veel lol van hebben en dat het maken van tekeningen nu een stuk gemakkelijker is.

Ron Siekman.





ALGEMENE GEGEVENS

De EF9345 is een goedkope semigrafische beeldscherm processor.
Hij is bedoeld voor het gebruik met een goedkope monochroom of kleuren monitor (64 us per lijn, 50 of 60 Hz refresh frekwentie).

De EF9345 geeft tot 25 regels van 40 karakters of 25 regels van 80 karakters.
De on-chip karakter generator beschikt over een 128 standaard, 5x7, karakterset en standaard semigraphische set.
Meer gebruikers definieerbare (8x10) alfanumerieke of semi grafische sets kunnen in het 16 Kb VDU-geheugen opgeslagen worden.

Deze set is alleen in de 40 koloms mode beschikbaar.

MICROPROCESSOR INTERFACE

De EF9345 heeft een 8-bits gemultiplexte adres/databus, microprocessor interface.
Deze is direkt compatibel met populaire (6801, 6805CT, 8048, 8051, 8035,...) microprocessors.

REGISTERS

De VDU-processor heeft 8 direkt toegankelijke registers :

- * R0 : commando/status register
- * R1, R2, R3 : Data registers
- * R4, R5 \ : Beide register paren wijzen naar het
- * R6, R7 / : VDU-geheugen.

Door deze registers wordt indirekt toegang tot het VDU-geheugen en 5 andere registers verkregen. :

- * ROR, DOR : Basis adres van de geheugen pagina die getoond wordt, en van externe karakter generatoren.
- * PAT, MAT, TGS : Gebruikt om de attributen en de formaten in te stellen, en om de timingsgenerator in te stellen.

VDU-GEHEUGEN

De gebruiker mag het VDU-geheugen tussen de volgende zaken verdelen :

- * Pagina's voor karaktercodes (2Kx8 of 3Kx8).
- * Externe karakter generators.
- * Algemeen gebruikers gebied.

Diverse soorten geheugencomponenten kunnen gebruikt worden.

* ROM, DRAM of SRAM.
* 2K x 8, 8K x 8, 16K x 4 organisatie.
* 500 ns cyclus tijd en 250 ns toegangs tijd is noodzakelijk.

40 KARAKTERS PER REGEL : KARAKTER CODE FORMAAT EN ATTRIBUTEN

Als de 40 karakter mode geselecteerd is, moet ook een van de drie karakter code formaten gekozen worden :

* 24-bit lang formaat :
  Bij elk karakter moeten de attributen meegegeven worden.

* 8/24-bit variabel formaat :
  Alleen bij attribuut wijziging moeten deze meegegeven worden.

* 16-bit kort formaat :
  Niet alle attributen zijn beschikbaar, moeten wel iedere keer meegegeven worden.

Het 16-bit korte formaat is compatibel met de EF9340/41 CRT controller.

Beschikbare karakter attributen :

- Achtergrond en voorgrond kleur (3 bits elk)
- Dubbel hoog, dubbel breed
- Knipperen
- inverteren
- onderstrepen
- verborgen
- invoegen
- accenten bij lowercase karakters
- 3x100 gebruiker definieerbare karakter generator in het geheugen
- 8 x 100 semigrafische 40-kleuren karakters.

80 KARAKTERS PER REGEL : KARAKTER CODE FORMAAT EN ATTRIBUTEN

Er zijn twee karakter code formaten beschikbaar :

* Lang (12-bits) met 4 attributen :

  - Knipperen
  - Onderstrepen
  - Inverteren
  - Kleuren

* kort (8-bits) : geen attributen

TIMINGS GENERATOR

De gehele timing wordt verzorgd door een 12 MHz klok.

De RGB output is 8 MHz bij de 40 karakter mode en 12 MHz bij de 80 karakter mode.

Daarnaast kan de gebruiker het volgende instellen :

* 50 of 60 Hz vertikale synchronisatie frequentie
* geinterlinieerd of niet
* gescheiden of samengestelde vertikale en horizontale synchronisatie output.

Verder is een samengestelde synchronisatie input aanwezig als een van de volgende zaken gewenst is :

* Een on-chip vertikale resynchronisatie
* Een on-chip groffe horizontale resynchronisatie
* Een off-chip 'high performance' horizontale resynchronisatie door het gebruik van een eenvoudige externe VCXO gecontrol-leerd door de on-chip fase vergelijker.

GEHEUGEN ORGANISATIE

LOGISCHE EN FISIEKE ADRESSERING

De fisieke 16 Kbyte geheugen ruimte is logisch verdeeld door de EF9345 in 40-byte buffers (figuur 1). Beter gezegd het logische adres wordt bepaald door het X, Y, Z trio, waarbij :

* X (= 0 tot 39) wijst naar een byte binnen een buffer
* Y (= 0, 1 ; 8 tot 31) wijst naar een buffer binnen een 1 Kb blok
* Z = (0 tot 15) wijst naar een blok

1 K=2^10=1024 kan niet precies door 40 gedeeld worden, daardoor bevat ieder blok 25 volledige buffers en een 24-byte overblijver. Aangenomen dat het fisieke geheugen een veelvoud van 2 Kbytes is, worden de overblijvers zodanig georganiseerd dat het volgende ver-kregen wordt :

* een volledige buffer (Y=0) in elk even blok
* een gedeeltelijke buffer (Y=1 ; X=32 tot 39) in elk oneven blok.

(Zie figuur 3)

POINTERS

Iedere X, Y en Z component van een logisch adres is binair gecodeerd en opgeslagen in twee 8-bits registers. Zo'n register paar is een pointer (figuur 2). De EF9345 heeft twee pointers :

* R4, R5 : hulp pointer
* R6, R7 : hoofd pointer

R5 en R7 hebben het zelfde formaat. Elk bevat een X component en de twee LSB's van een Z component. Deze opslag zorgt voor een verdeling van Z in 4 districten van 4 blokken ieder.

R5 en R7 wijzen ieder naar een bloknummer in een district. R4 en R6 hebben een verschillend formaat : beide bevatten een Y component en de LSB van het district nummer, maar R6 bevat de MSB's van beide districten.

Figuur 3 toont de logische naar de fisieke adres omzetting, zoals dit in de chip gebeurt.

DATASTRUKTUUR IN HET GEHEUGEN

Een pagina is een stuk geheugen dat weergegeven kan worden op een beeldscherm tot 25 regels karakters. Overeenstemmend met het karakter code formaat, komt elke regel overeen met 2 (of 3) 40-bytes buffers. Deze set van 2 (of 3) buffers vormen een regel bugger (figuur 1). De buffers die tot een regel buffer behoren, moeten aan de volgende voorwaarden voldoen :

* ze hebben het zelfde Y adres
* ze hebben het zelfde district nummer
* ze liggen in 2 (of 3) opvolgende (modulo 4) blok adressen in hun eigen district.

Dus, een regel buffer wordt bepaald door zijn eerste buffer adres en formaat.
Een pagina is een set van opvolgende regel buffers :

* met het zelfde formaat
* met het zelfde district nummer
* met het zelfde blok adres van de eerste buffer. Dit blok adres moet even zijn.
* met opeen volgende (modulo 24) Y adressen.

Een gevolg hiervan is dat een pagina niet over een districts grens mag lopen. Algemeen gebruikt geheugen mag gebruikt worden, maar ze moeten wel voldoen aan de buffer of regel buffer structuur. Zie figuur 2 voor de pointer verhoging als gevolg van deze data structuur.

GEHEUGEN TIJD DELING (zie figuur 4)

De geheugen interface heeft een 5ØØ ns tijdcyclus nodig. Dat is een 2 Mbyte/s geheugen bandbreedte. Deze bandbreedte wordt gedeeld tussen :

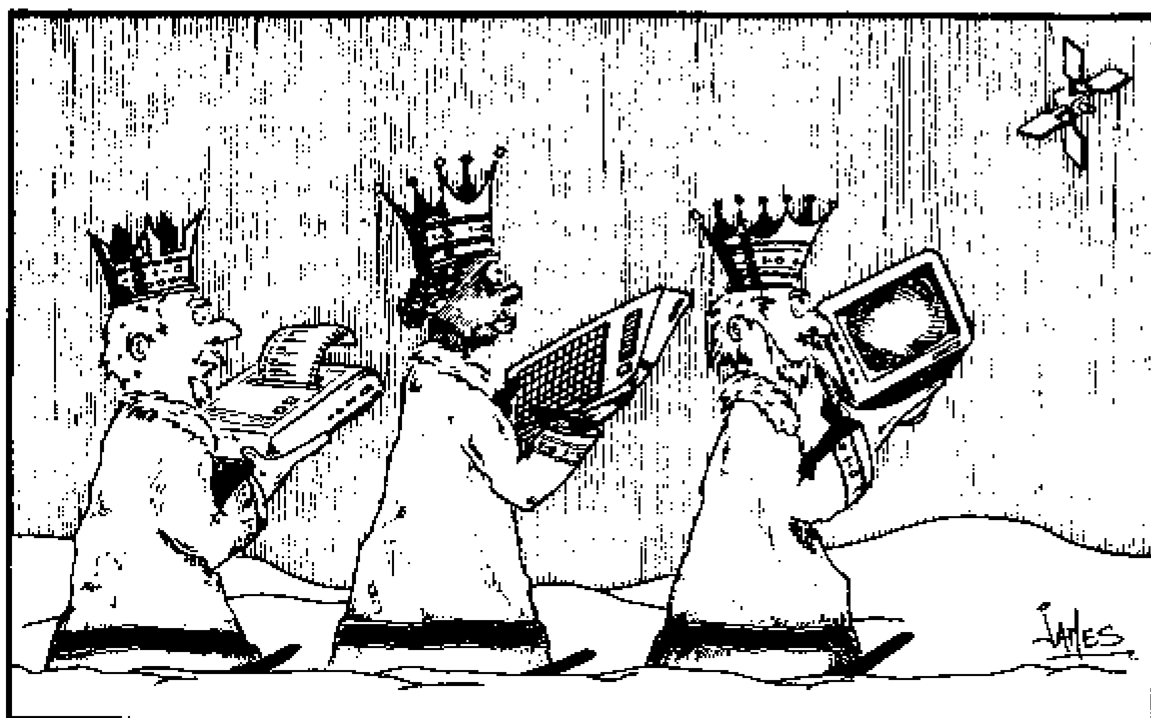
* het lezen van een rijbuffer vanuit het geheugen om de interne rijbuffer te laden (tot 12Ø bytes per rij).
* het lezen van gebruiker gedefinieerde karakter delen vanuit het geheugen (1 byte per microseconde).
* indirekte processor lees en schrijf opdrachten.
* refresh tijd voor DRAM.

Een vast plaatsings schema bepaald de tijdsdeling.

Opmerkingen bij figuur 4

1. Dummy cycli zijn lees opdrachten op dummy adressen
2. Refresh cycli zijn lees cycli veroorzaakt door een 8-bit automatisch verhogende teller. Het lage byte van de adressen ADM(Ø:7) cycli tot de volledige 256 adressen vergt minder dan 1 ms.
3. De processor heeft iedere microseconde indirekte toegang tot het geheugen, behalve tijdens de eerste en de laaste lijn van een rij, als de interne buffer opnieuw geladen moet worden.

Tijdens deze lijnen heeft de processor gedurende 1Ø4 us geen toegang tot het geheugen. Dit is ook het geval als er geen gebruiker gedefinieerde karakter delen geadresseerd zijn.



FIGUUR 1 - GEHEUGEN REGEL BUFFER

X

0 31 39

DISTRIKT

BLOK Ø

BLOK 1

BLOK 2

BLOK 3

Y

DISTRIKT

DISTRIKT

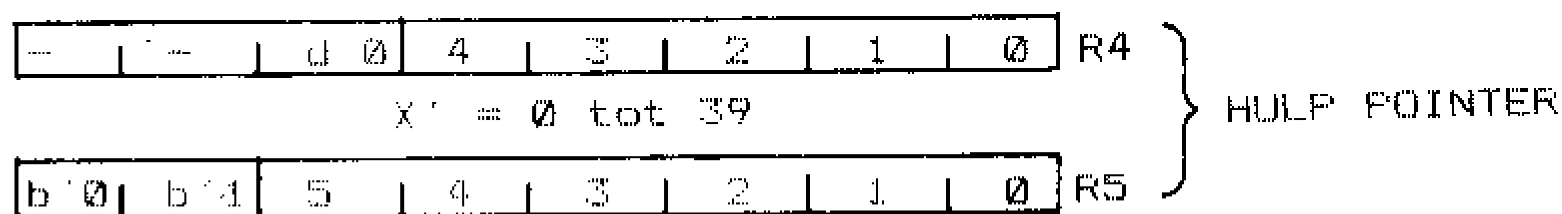
120-BYTE RIJ BUFFER   8Ø-BYTE RIJ BUFFER

* regelbuffers liggen in een distrikt
* twee of drie opvolgende blok adressen (modulo 4)
* eerste blok adres is even

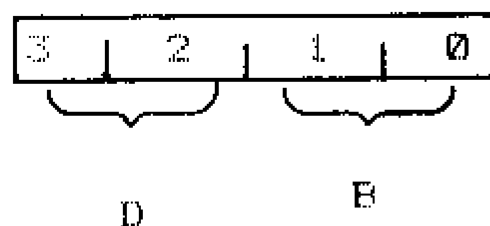
Y = (Ø,1 ; 8 tot 31)

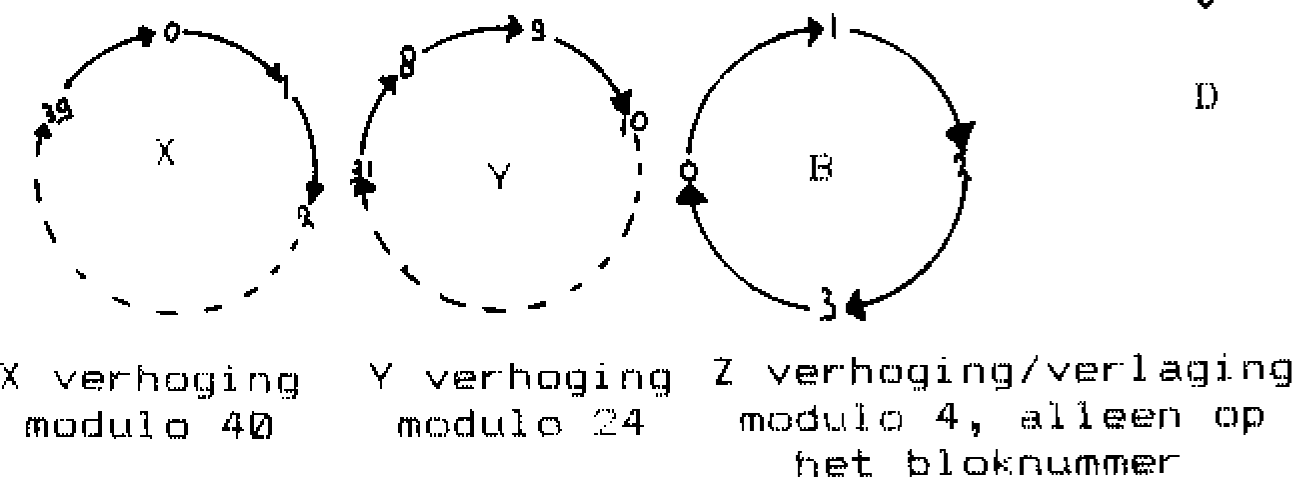


Y' = (Ø,1 ; 8 tot 31)



Z = (Ø tot 15)



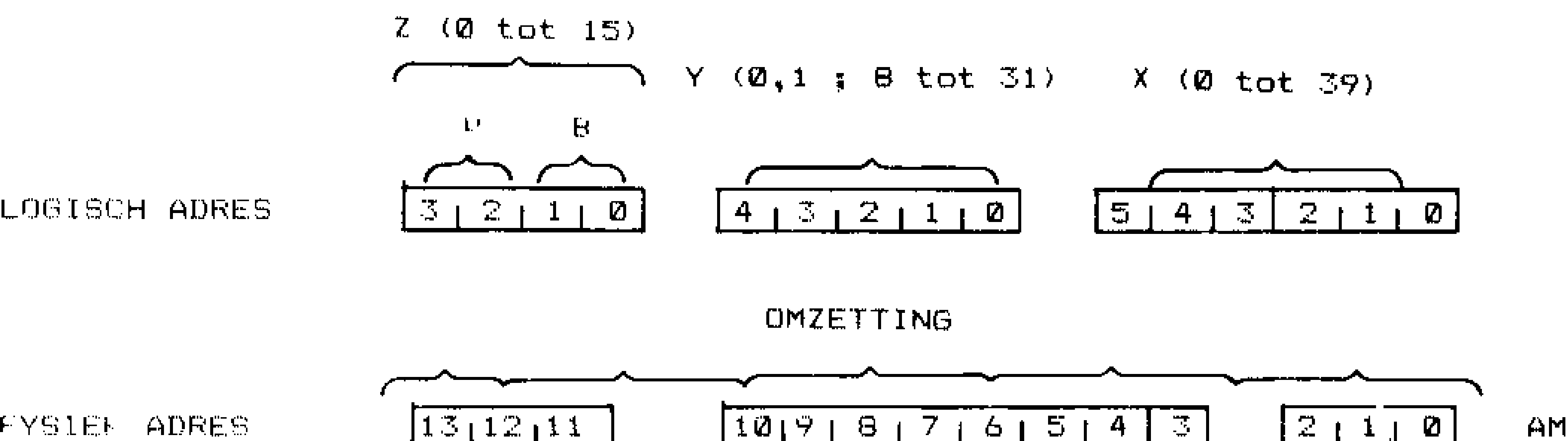


X verhoging modulo 4Ø

Y verhoging modulo 24

Z verhoging/verlaging modulo 4, alleen op het bloknummer

FIGUUR 2 - AUTOMATISCHE POINTER VERHOGING

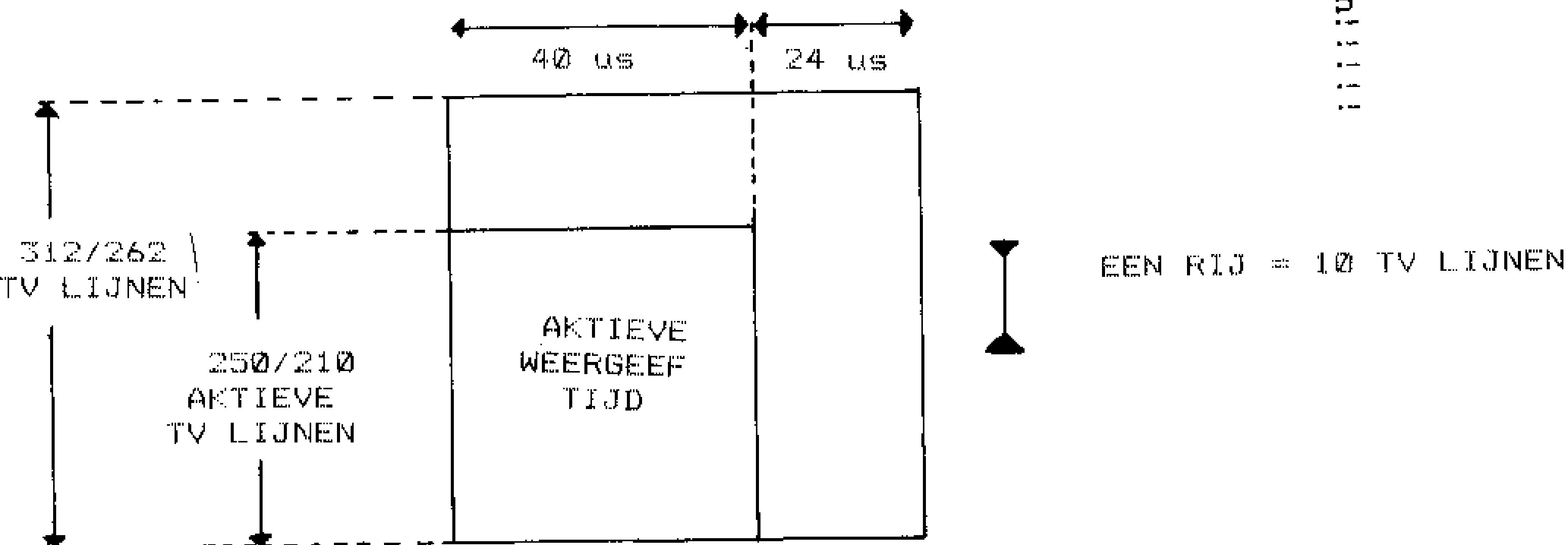


X en Y CONDITIE			FYSIEK ADRES AM (3:10)							
			10	9	8	7	6	5	4	3
Y >= 8	X5 = 0		b0	Y4	Y3	Y2	Y1	Y0	X4	X3
	X5 = 1		b0	0	0	Y2	Y1	Y0	Y4	Y3
Y < 8	Y0 = 0		b0	0	0	X5	X4	X3	0	0
	Y0 = 1	b0 = 0	X3	0	0	1	X5	X4	0	0
		b0 = 1	1	0	0	1	X5	X4	0	0

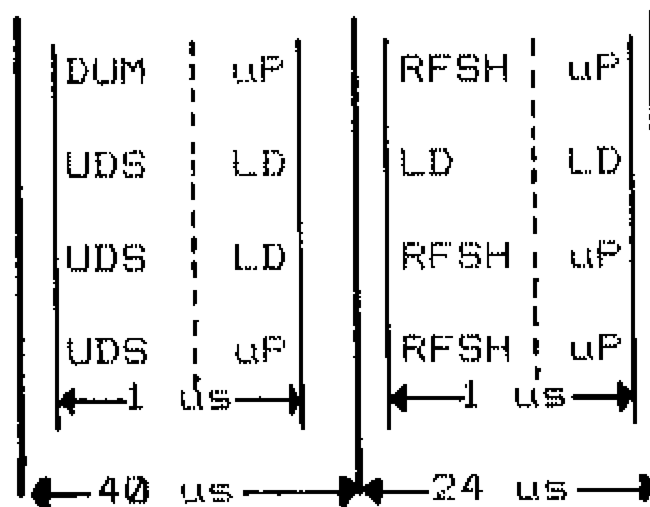
FIGUUR 3 - LOGISCHE NAAR FYSIEKE ADRES OMZETTING

Zie pag 3 onderaan!!!!!

FIGUUR 4 - GEHEUGEN CYCLUS PLAATSING



INAKTIEVE LIJNEN	DUM	uP	RFSH	uP
LAATSTE RIJ LIJN	UDS	LD	LD	LD
EERSTE RIJ LIJN	UDS	LD	RFSH	uP
OVERIGE RIJ LIJNEN	UDS	uP	RFSH	uP



GEHEUGEN CYCLUS
DUM: dummy cyclus
uP: indirekte toegang
RFSH: refresh cyclus
UDS: gebruiker gedefinieerde karakterdeel leescyclus
LD: leescyclus voor laden interne rij buffer

*** GMSnew ***

De nieuwe versie van GMS bevat enkele aanvullingen met betrekking tot de eerder gelanceerde versie (A.N. jrg.5, nr.4)

- DISK gebruik
  Bij aanroep van het SAVE of LOAD onderdeel verschijnt een CATalog van de disc, waarna een filenaam ingetikt mag worden.
  Als er tijdens er tijdens laden en/of schrijven iets fout gaat, verschijnt de catalog overnieuw.
- extra functies van 'potlood'
  Nadat voor het potloodje (met gum) is gekozen kan men door het indrukken van een lettertoets gebruik maken van de volgende functies:

  [M] Move
  'bewaar' huidige potlood-positie.

  [D] Draw
  trek een lijn vanaf laatst bewaarde punt en bewaar eindpunt van getekende lijn.

  [C] Circle
  teken een cirkel met het middelpunt in de laatst bewaarde positie en een straal die gelijk is aan de afstand tussen laatst bewaarde positie en huidig punt.

  [F] Fill
  Vul de aan te geven posities met een te kiezen paint-patroon (GAGS-ROM noodzakelijk !).
  Er verschijnt een keuzemenu met een aantal paint-patronen: kies gewenste patroon en laat de cursorpijl BOVEN buiten beeld verdwijnen.
  De cursor-pijl verschijnt weer en kan gebruikt worden om de te vullen patronen aan te wijzen.
  Als het vullen een onbevredigend resultaat geeft (b.v. lekje), dan kan de actie ongedaan gemaakt worden door onder buiten beeld te verdwijnen.

Van de nieuwe versie van GMS (weet u nog: Grafisch Manipulatie Systeem) bestaat slechts een uitvoering: deze is geschikt voor ATOMS met geheugen vanaf #2800 tot #A000.

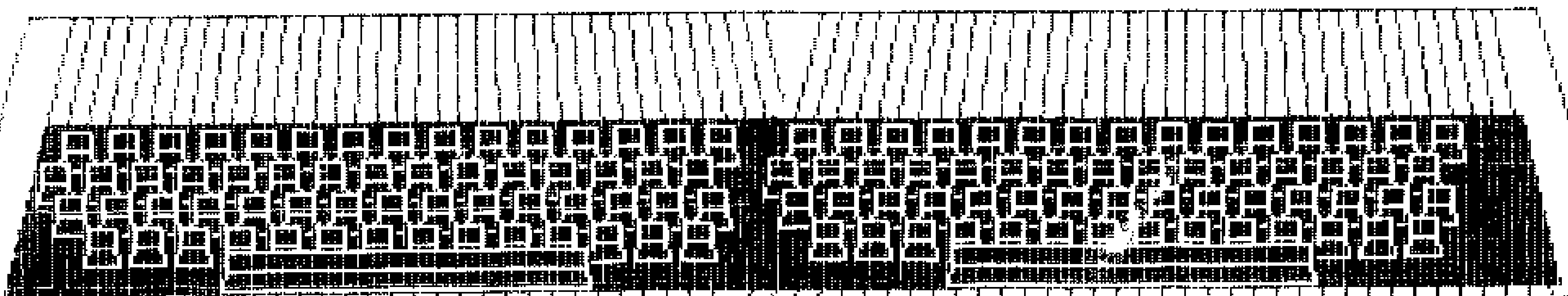
FILES:

GMSNW.A	:assemblersource
GMSNW.B	:'binaire'-routines (afkomstig van GMSNW.A)
GMSNW.M	: hoofdprogramma (BASIC)
GMSNrun	: kort opstart-programma
SCREEN	: introductiescherm


Andre de Bruin,
Baljuw 11,
2671 HL Naaldwijk.

G.M.S. G.M.S.

'Met het menu van G.M.S.
gaat er een wereld open.'



DUOTASK

We weten allemaal dat de automatiseringswereld niet alleen uit ATOMs bestaat. Zo zijn er op de markt nogal wat MINI- en MAINFRAME-computers die, hoewel ze slechts een processor bezitten, toch meerdere programma's tegelijkertijd kunnen verwerken. Deze mogelijkheid wordt aangeduid als 'time-sharing'.
De meest eenvoudige vorm hiervan is 'time-slicing': elk programma krijgt telkens een 'plakje' processortijd.
Hierna volgt de beschrijving van het DUOTASK programma: een programma dat het mogelijk maakt om twee programma's tegelijkertijd op de ATOM te gebruiken. (daarbij is DUOTASK zelf NIET meegerekend !!)

Als dit programma'tje opgestart is, dan kunnen we de ATOM als voorheen normaal gebruiken: we kunnen een BASIC programma laden, waarna we met RUN het programma starten.
Als we dan echter tegelijkertijd op de CTRL en SHIFT toets drukken, dan merken we dat het BASIC programma stopt. Nu kunnen we een ander BASIC programma laden, dit moet dan natuurlijk niet het al aanwezige programma vernietigen.(Met *LOAD kunnen we het programma naar een vrij gebied in het geheugen dirigeren.)

Het laatst aan boord getrokken programma kunnen we ook laten RUNnen, waarna we voor de grap tegelijkertijd de CTRL en SHIFT toets indrukken.
Tot onze niet geringe verbazing zullen we dan merken dat het laatst geactiveerde programma nu 'hangt' (stopt), terwijl het eerder genoemde BASIC programma weer doorgaat, daar waar hij gestopt was.
We kunnen nu naar believen CTRL-REPT en CTRL-SHIFT indrukken en zo dan weer het ene en dan weer het andere programma de beurt geven.

Leuk wordt het pas als we op CTRL-SHIFT-REPT drukken , we merken dan namelijk dat beide programma's tegelijkertijd werken !!. (tip: eerst de CTRL-toets,daarna pas de anderen en in omgekeerde volgorde weer loslaten)

VOORBEELD:
{eerst het DUOTASK programma RUNnen}
VOER IN:

```
?18=#29;NEW  { dus het programma begint op adres #2900 }
10 PRINT"EEN NIEUWE LENTE "
20 GOTO 10

RUN
```

Het scherm stroomt nu over met literaire tekst. Voer daarna de volgende handelingen uit:

```
CTRL-REPT
?18=#30;NEW
 10 PRINT $7
 20 GOTO 10

RUN
CTRL-SHIFT-REPT
```

(een echte bende krijg je pas als je regel 10 verandert in:
10 PRINT" EEN NIEUW GELUID")

EEN TIP VAN DE SLUIER

Als het DUOTASK programma wordt gestart, dan verplaatst deze de eerste 4 geheugenpagina's (adress #0..#3FF) naar een andere plaats in het geheugen.

Die eerste vier pagina's bevatten alle systeemvariabelen en de stack van de 6502.

Die 'systeemvariabelen' bevatten alle informatie van een lopend programma (startadres van programma, welk statement wordt op dit moment uitgevoerd etc.)

Als we nu op CTRL-SHIFT of CTRL-REPT drukken, dan worden de beide genoemde systeempagina's weer uitgewisseld.

Een ding is echter nog niet verteld: het programma die de wisseling uitvoert, moet constant naar het toetsenbord gluren om te zien of er al op CTRL-SHIFT, CTRL-REPT of CTRL-SHIFT-REPT is gedrukt.
Welnu, dit is opgelost door dat programma als 'interrupt'-routine te programmeren:

De VIA wordt zo geprogrameerd dat deze op gezette tijden een interrupt doorgeeft aan de processor.
Als de processor een interrupt krijgt, dan begint deze de routine ':KK7' (zie programma) uit te voeren.Die routine wordt slechts een maal aangeroepen: hij copieert de huidige eerste 4 geheugenpagina's naar een vrije plaats in het geheugen, maar zorgt er ook voor dat volgende interrupts door routine 'KK2' afgehandeld worden.
Die routine ':KK2' (Die dus bij elke interrupt weer aangeroepen wordt), gluurt naar het toetsenbord om er achter te komen of een van de speciale toetsvolgorden (zoals CTRL-SHIFT) al is ingedrukt.
Als dit zo s, dan beslist de routine of de beruchte 4 pagina's gewisseld moeten worden.

Als we een keer op CTRL-SHIFT-REPT gedrukt hebben, dan wordt bij elke volgende aanroep een wisseling uitgevoerd, zodat de beide BASIC programma's dus 'tegelijk' werken.
(OPMERKING: voor lezers van ACORN USER/ATOM FORUM: het is dus NIET zo dat CTRL-SHIFT-REPT ingedrukt moet blijven, de redacteur van die rubriek is wat dat punt betreft de fau d in gegaan)

HET DUOTASK PROGRAMMA

Het DUOTASK programma kan gewoon vanaf adres #2900 geladen worden #2900.

Als het programma dan gestart wordt, dan vraagt deze waar de machinecode geplaatst mag worden. Als je een standaard 12Kb RAM -ATOM hebt, is #3600 een fijn antwoord.
Vervolgens vraagt het programma om een vrij stuk geheugen van 1KB, met de standaardATOM is #3800 aan te raden (precies 1KB voor #3C00).
Als laatste vraagt het programma of je 'locale' of 'globale' variabelen wil gebruiken.
Het verschil? wel als je 'locale' variabelen kiest, dan gebruiken beide programma's verschillende variabelen.(vergelijk: 'globaal' en 'locaal' bij P-charme)

In praktijk is het vaak makkelijk om 'locale' variabelen te gebruiken, daar we dan twee totaal onafhankelijke BASIC-programma's hebben. Als we echter de twee programma's met elkaar willen laten 'spreken', dan moeten we 'globale' variabelen kiezen.

OPMERKING: in het bovenstaande verhaal wordt telekenmale gesproknen over twee (BASIC) programa's die tegelijkertijd kunnen werken, maar in feite werken er 3 naast elkaar:

- -Het DUOTASK programma zelf
- -Twee 'andere' programma's

Om niet te veel verwarring te stichten heb ik de twee 'anderen', in het verhaal 'BASIC'-programma's genoemd, dit mogen echter ook machinetaal programma's zijn.
Zo is het bijvoorbeeld mogelijk om de teksteditor aan te roepen, en daarnaast nog een programma te plaatsen dat een melodie'tje speelt.

Het DUOTASK principe is in feite toepasbaar bij welk schakelsysteem, box, geheugenkaart dan ook.
Problemen treden op (zoals we in het voorbeeld ook zagen) als twee programma's hetzelfde geheugengebruiken.Het beeldschermgeheugen geef wat dat betreft de meeste last. Daardoor is het dus niet mogelijk om twee LISP-programma's naast elkaar te laten draaien (een gedeelte van de LISP-'compiler' bevindt zich vanaf adres #8200)



Andre de bruin
Baljuw 11
2671 HL Naaldwijk

```
 10 REM ***               DUOTASK          ***
 20 REM *** made by: Andre de Bruin        ***
 30 REM *** Baljuw 11, NAALDWIJK, HOLLAND ***
 40 REM ***                                ***
 50  @=0;DIM VV16,KK16; FOR I=1TO16;VVI=#2800;KKI=#2800;NEXT
 60  PRINT"WHERE CODE ?"'" FIRST BYTE AFTER PROGRAM =";@=0
 70  PRINT&?#23+?#24*#100; INPUT W
 80  INPUT'" WHERE SECOND ZEROPAGE "Z
 90  INPUT'"VARIABLES  : "'"   1=LOCAL , 0=GLOBAL",V
100  PRINT$21; FOR I=1TO2;P=W
110  B=KK7%256;C=KK7/256;D=KK2%256;E=KK2/256
120[    \{ PROGRAM THE 6522 VIA TO GENERATE INTERRUPTS }
130     \{ THE INTERRUPT VECTOR POINTS TO 'KK7'          }
140 :KK0  LDA@#40;STA#B80B;LDA@#C0;STA#B80E;LDA@#50;STA#B804
150       LDA@#FF;STA#B807;STA#B805;LDA@B;STA#204
160       LDA@C; STA#205;CLI;RTS
170
180     \{ CHANGE INTERRUPT-VECTOR: NOW POINT TO ':KK2'      }
190 :KK7  LDA#B804; LDA@D;STA#204;LDA@E;STA#205
200       LDA@1;STA#23F;LDA#B000;PHA
210       TXA;PHA;TYA;PHA;TSX;STX #23E; LDX@0;:KK6;LDA 0,X
220       STA Z,X ,X;LDA #100,X;STA Z+#100 ,X
230       LDA#200,X;STA Z+#200,X;LDA#300,X;STA Z+#300,X
240       INX;BNE KK6;LDA@2;STA#23F;JMP VV9
250
260 :KK2  LDA#B804
270       LDA#B001;CMP@63;BEQ KK4  { IS ctrlshift PRESSED ?}
280       CMP@191;BEQ KK3           { IS ctrl PRESSED ?}
290       JMP KK16
300 :KK3  LDA#B002;AND@64;BEQ KK15
310 :KK16 LDA VV11;CMP@3;BEQ KK12;JMP VV10
320 :KK15 LDA@1;STA VV11;CMP#23F;BNE KK12;JMP VV10
330 :KK4  LDA#B002;AND@64;BEQ KK14
340       LDA@2;STA VV11;CMP#23F;BNE KK12;JMP VV10
350 :KK14 LDA@3;STA VV11 \ { CTRL-SHIFT-REPT WAS PRESSED }
360 :KK12 LDA#B000;PHA; TXA;PHA;TYA;PHA;TSX;STX#23E
370       LDX@0;:KK8;LDA 0,X;STA VV12;LDA Z,X;STA 0,X;LDA VV12
380       STA Z,X; LDA#100,X;STA VV12;LDA Z+#100 ,X;STA#100,X
390       LDA VV12 ;STA Z+#100,X
400       LDA#200,X;STA VV12;LDA Z+#200 ,X;STA#200,X
410       LDA VV12;STA Z+#200,X
420]; IF V=1 GOTO g ;REM {if "global" selected, GOTO g}
430[;     CPX@#22;BPL KK9;JMP KK10
440 :KK9  CPX@#8D;BMI KK11 ; ]
450g[:KK10 LDA#300,X;STA VV12;LDA Z+#300 ,X;STA#300,X
460       LDA VV12;STA Z+#300,X
470 :KK11 INX;BNE KK8
480 :VV9  LDX#23E;TXS; PLA;TAY;PLA;TAX; PLA;STA#B000
490 :VV10 LDA#23F;STA#8000;PLA;RTI
500 :VV11 NOP \{"mode": 1=proces1 2=proces2 3=both }
510 :VV12 NOP
520]; NEXT;PRINT$6
530 PRINT"AFTER break RESTART "'"WITH LINK #"&W'
540 PRINT"TOP OF PROGRAM: #"&VV12+1,'"USED ADRESSES :"'
550 PRINT" #23F  : IDENTIFICATION (1 OR 2)"'
560 PRINT" #23E  : STACKPOINTER"'
570 PRINT" "&VV11"  : STATE"'" ",&VV12"  : SWAPBYTE "'
580 LINK W;END
```

LMOVE (Line-MOVE)

Syntax: LMOVE lnum1,lnum2,lnum3 (R)

Met dit commando kan men regels binnen een programma verplaatsen. De regels lnum1 t/m lnum2 worden verplaatst naar lnum3 waarbij alle verplaatste regels lnum3 als regelnummer krijgen. Als lnum3 reeds bestaat wordt het blok met regels direct hierna tussengevoegd. Lnum3 zelf mag zich niet binnen het te verplaatsen blok bevinden en de gebruiker wordt hiertegen beschermd middels een ERROR 174. Het programma zal nadat een move-operatie heeft plaatsgevonden exact even lang zijn als voorheen. Ook tijdens het move-proces zal het programma nooit langer zijn.

Verder is een renumber-optie mogelijk door het commando af te sluiten met de letter R. Na het voltooien van het move-proces wordt het programma dan in z'n geheel hernummerd, beginnend met regelnummer 10 en met stappen van 10.

Het LMOVE-commando is in de eerste plaats bedoeld voor assembler source-programma's. Daar komt het bijvoorbeeld nog al eens voor dat er met subroutines moet worden geschoven om een optimale indeling te verkrijgen. Bij normale BASIC-programma's moet de nodige voorzichtigheid in acht genomen worden omdat LMOVE geen rekening houdt met regelnummers in GOTO's, GOSUB's e.d.

Voor de duidelijkheid volgen nu enkele voorbeelden. Elk voorbeeld heeft het origineel als uitgangs-situatie.

ORIGINEEL	LMOVE 20,40,75	LMOVE 60,90,20R	LMOVE 0,30,99
10 aaaaa	10 aaaaa	10 aaaaa	40 ddddd
20 bbbbb	50 eeeee	20 bbbbb	50 eeeee
30 ccccc	60 fffff	30 fffff	60 fffff
40 ddddd	70 ggggg	40 ggggg	70 ggggg
50 eeeee	75 bbbbb	50 hhhhh	80 hhhhh
60 fffff	75 ccccc	60 ccccc	99 aaaaa
70 ggggg	75 ddddd	70 ddddd	99 bbbbb
80 hhhhh	80 hhhhh	80 eeeee	99 ccccc

LCOPY (Line-COPY)

Syntax en gebruik zijn hetzelfde als bij LMOVE. Het resultaat is echter in die zin verschillend dat de "verplaatste" regels op de oorspronkelijke plaats niet verwijderd worden. In tegenstelling tot LMOVE wordt het programma nu dus wel langer.

Opmerkingen:

De subroutines in het AXXX-gebied hebben betrekking op P-Charme 173. Het "¦"-teken is een ge-inverteerde "\" (logische OR). Bij gebruik van de renumber optie kunnen er vreemde dingen gebeuren. Toevoeging van de volgende 2 regels verhelpen dit euvel:

```
1061 LDA @#A3  \ table #A3xx !
1062 STA #94   \ !
```

LITERATUUR

P-CHARME UITBREIDEN AN 2 1984 blz 36..37
P-FUN AN 7 1984 blz 88..93

```
  10  PROGRAM LMOVE.STAT & LCOPY.STAT
  20
  30  PRINT $21;GOSUB a
  40  GOSUB a;PRINT $6
  50  A=end;T=P-3
  60  END
  70
  80 aREM 6502 SOURCE (MINIAS)
  90 [
 100   .BA A
 110
 120 :ptr    = #58
 130 :length = #97
 140 :buffer = #140
 150
 160 :lmove
 170   CLC
 180   BCC lcopy+1
 190 :lcopy
 200   SEC
 210   PHP
 220   LDX #06      \ direct mode ?
 230   DEX
 240   BNE error
 250   STX #95      \ disable find
 260   LDX @3
 270   STX length
 280 :nxt'lnum
 290   JSR #A4D6    \ get lnum
 300   BCC error
 310   DEC length
 320   BEQ lnum'ready
 330   JSR #C231    \ test comma
 340   BCC nxt'lnum
 350 :error
 360   JMP #C2AC    \ error 174
 370
 380 :lnum'ready
 390   JSR #F291    \ get char
 400   PLP
 410   EOR @CH"R"   \ renum ?
 420   PHP
 430   BEQ P+3
 440   DEY
 450   JSR #C4E7    \ test end stat
 460
 470   LDA #16+2    \ C=0
 480   STA buffer+1
 490   SBC #16+1    \ if L3>L2
 500   LDA #25+2    \ then ok
 510   STA buffer
 520   SBC #25+1
 530   BCS begin
 540   LDA #16+0    \ if L3<L1
 550   SBC #16+2    \ then ok
 560   LDA #25+0
 570   SBC #25+2
 580   BCC error
 590
 600 :begin
 610   LDX @1
 620   STX #04
 630   JSR #A124    \ find L1
 640   BCS ready    \ if L1>L2 ready
 650
 660   DEY          \ X=0
 670 :nxt'char
 680   INY          \ L1 to buffer
 690   INX
 700   LDA (ptr),Y
 710   STA buffer+1,X
 720   CMP @#0D
 730   BNE nxt'char
 740   INX
 750   STX length
 760
 770   PLP
 780   PHP
 790   BCS skp'del
 800   LDA ptr
 810   JSR #A1BF    \ delete L1
 820 :skp'del
 830   JSR #A12D    \ get next L1
 840
 850   LDX @2
 860   JSR #C632    \ find L3
 870   BCS no'lnum3
 880 :nxt'lnum3
 890   DEY
 900   JSR #C63B    \ skip all L3's
 910   BCC nxt'lnum3
 920
 930 :no'lnum3
 940   DEY
 950   STY #96      \ Y=1
 960   DEY
 970   JSR #AEBF    \ insert buffer
 980   BCS begin    \ branch always
 990
1000 :ready
1010   PLP
1020   BEQ renum
1030   JMP #C2CA    \ back to basic
1040
1050 :renum
1060   STX #03      \ X=0
1070   JMP #A282    \ renumber
1080 :end
1090
1100   .BA T
1110   .AS "LMOVE"
1120   .DB lmove!#B000
1130   .AS "LCOPY"
1140   .DB lcopy!#B000
1150   .BY #80
1160   .WO end
1170 ]
1180  RETURN
```

```
 10  PROGRAM WORD.STAT & DBYTE.STAT
 20
 30  PRINT $21;GOSUB a
 40  GOSUB a;PRINT $6
 50  A=end;T=P-3
 60  END
 70
 80 aREM 6502 SOURCE (MINIAS)
 90 [
100   .BA A
110
120 :getexpr  = #C7BB
130 :asign    = #C4DE
140 :to'zp52  = #C3CB
150
160 :word
170   CLC
180   BCC dbyte+1
190 :dbyte
200   SEC
210   PHP
220   JSR getexpr
230   JSR asign
240   DEX
250   LDA #16,X
260   LDY #25,X
270   PLP
280   BCC skpxchg
290   TAY
300   LDA #25,X
310 :skpxchg
320   PHA
330   TYA
340   PHA
350   JSR to'zp52
360   LDY @1
370 :nxtbyt
380   PLA
390   STA (#52),Y
400   DEY
410   BPL nxtbyt
420   JMP #C55B
430 :end
440
450   .BA T
460   .AS "WORD";.DB word|#8000
470   .AS "DBYTE";.DB dbyte|#8000
480   .BY #80;.WO end
490 ]
500  RETURN
510
520  syntax:  WORD expr = expr
530           DBYTE expr = expr
540
550  ofwel:   WORD adres = data
560           DBYTE adres = data
570
580  WORD A=V  ->  ?A=V%256;A?1=V/256
590  DBYTE A=V  ->  ?A=V/256;A?1=V%256
```

```
 10  PROGRAM WORD.FUN & DBYTE.FUN
 20
 30  PRINT $21;GOSUB a
 40  GOSUB a;PRINT $6
 50  A=end;T=P-3
 60  END
 70
 80 aREM 6502 SOURCE (MINIAS)
 90 [
100   .BA A
110
120 :getfact  = #C8BC
130 :to'zp52  = #C3CB
140 :to'stack = #C97C
150
160 :word
170   CLC
180   BCC dbyte+1
190 :dbyte
200   SEC
210   PHP
220   JSR getfact
230   JSR to'zp52
240   LDY @0
250   LDA (#52),Y
260   INY
270   PLP
280   BCS skpxchg
290   LDA (#52),Y
300   DEY
310 :skpxchg
320   PHA
330   LDA (#52),Y
340   JSR to'stack
350   PLA
360   STA #24,X
370   RTS
380 :end
390
400   .BA T
410   .AS "WORD"
420   .DB word|#8000
430   .AS "DBYTE"
440   .DB dbyte|#8000
450   .BY #80
460   .WO end
470 ]
480  RETURN
490
500
510
520  syntax:  WORD fact
530           DBYTE fact
540
550  ofwel:   WORD adres
560           DBYTE adres
570
580  V=WORD A  ->  V=?A+A?1*256
590  V=DBYTE A  ->  V=?A*256+A?1
```

```
10  PROGRAM BIT.STAT
20
30  PRINT $21;GOSUB a
40  GOSUB a;PRINT $6
50  A=end;T=P-3
60  END
70
80 aREM 6502 SOURCE (MINIAS)
90 [
100   .BA A
110
120 :getfact  = #C8BC
130 :getexpr  = #C78B
140 :tstcomma = #C231
150 :asign    = #C4DE
160 :to'zp'y  = #C3CD
170 :updbit   = #F7A1
180
190 :bit
200   JSR getexpr
210   JSR tstcomma
220   JSR getexpr
230   JSR asign
240   DEX
250   LDA #16,X
260   ORA #25,X
270   ORA #34,X
280   ORA #43,X
290   BEQ zero
300   LDA @1
310 :zero
320   STA #5E
330   LDY @#5F
340   JSR to'zp'y
350   DEC #04
360   LDA #15,X
370   EOR @#FF
380   JSR updbit
390   JMP #C55B
400 :end
410
420   .BA T
430   .AS "BIT"
440   .DB bit!#8000
450   .BY #80
460   .WO end
470 ]
480  RETURN
490
500
510  syntax:  BIT fact,expr = expr
520
530  ofwel:   BIT bitnr,adres = data
540
550  BIT 5,A=1   ->   ?A=?A!#20
560
570  BIT N,A=0   ->   ?A=?A&%(2^N+.1):#FF
580
590
```

```
10  PROGRAM BIT.FUN
20
30  PRINT $21;GOSUB a
40  GOSUB a;PRINT $6
50  A=end;T=P-3
60  END
70
80 aREM 6502 SOURCE (MINIAS)
90 [
100   .BA A
110
120 :getfact  = #C8BC
130 :tstcomma = #C231
140 :to'zp52  = #C3CB
150 :to'yreg  = #CF41
160 :to'stack = #C97C
170 :bitmask  = #F7C9
180
190 :bit
200   JSR getfact
210   JSR tstcomma
220   JSR getfact
230   JSR to'zp52
240   JSR to'yreg
250   TYA
260   EOR @#FF
270   AND @#07
280   TAY
290   LDA bitmask,Y
300   LDY @0
310   AND (#52),Y
320   BEQ zero
330   LDA @1
340 :zero
350   JMP to'stack
360 :end
370
380   .BA T
390   .AS "BIT"
400   .DB bit!#8000
410   .BY #80
420   .WO end
430 ]
440  RETURN
450
460
470  syntax:  BIT fact,fact
480
490  ofwel:   BIT bitnr,adres
500
510  V=BIT 5,A   ->   V=(?A&#20=#20)
520
530  V=BIT N,A   ->   V=(?A&%(N^2+.1)
540                      =%(N^2+.1))
```

```
 10  PROGRAM PBIN.STAT
 20
 30  PRINT $21;GOSUB a
 40  GOSUB a;PRINT $6
 50  A=end;T=P-3
 60  END
 70
 80 aREM 6502 SOURCE (MINIAS)
 90 [
100   .BA A
110
120 :getexpr = #C78B
130 :prchar  = #CA4C
140 :print   = #C334
150 :to'yreg = #CF41
160
170 :pbin
180   JSR getexpr
190   JSR to'yreg
200   TYA
210   LDX @8
220 :nxtbit
230   ASL A
240   PHA
250   LDA @CH"0"
260   ADC @0
270   JSR prchar
280   PLA
290   DEX
300   BNE nxtbit
310   JMP print
320 :end
330
340   .BA T
350   .AS "PBIN"
360   .DB pbin!#8000
370   .BY #80
380   .WO end
390 ]
400  RETURN
410
420
430  syntax:  PBIN expr <PRINT>
440
450  PBIN #8A        ->  10001010
460
470  PBIN 2" two"'   ->  00000010 two
480
490  dus: drukt laagste byte van
500       de expressie af in de
510       vorm van een 8-bits
520       binair getal en gaat
530       verder alsof het een
540       PRINT opdracht betrof.
550
560
570
580
590
```

```
 10  PROGRAM BIN.FUN
 20
 30  PRINT $21;GOSUB a
 40  GOSUB a;PRINT $6
 50  A=end;T=P-3
 60  END
 70
 80 aREM 6502 SOURCE (MINIAS)
 90 [
100   .BA A
110
120 :to'stack = #C97C
130 :getnsp   = #F291
140
150 :bin
160   LDX #04
170   LDA @0
180   JSR to'stack
190 :nxtbit
200   JSR getnsp
210   LSR A
220   EOR @CH"0"/2
230   BNE endbin
240   ROL #15,X
250   ROL #24,X
260   ROL #33,X
270   ROL #42,X
280   JMP nxtbit
290 :endbin
300   STY #03
310   RTS
320 :end
330
340   .BA T
350   .AS "BIN"
360   .DB bin;#8000
370   .BY #80
380   .WO end
390 ]
400  RETURN
410
420
430  syntax:  BIN bits (max 32)
440
450  V=BIN 101          ->  V=5
460
470  V=BIN 1100 1000    ->  V=#C8
```

LET OP: DEZE TRUUK ALLEEN TOEPASSEN ALS JE BANDJE HELEMAAL LEEG IS.

Na een hele tijd (+/- 2 jaar) nadat mijn computer is gekocht, heb ik hem verbouwd naar mijn wensen.
Ik ben nu in het bezit van :Eprommer (zero)
omgebouwde typemachine tot printer
32K battery-backup (zelfbouw)
schakelkaart voor 8*2732 (made in Heytse)
omgebouwde tv tot "monitor"
MDCR voor data opslag

Het gebruik van de MDCR valt 100% mee in vergelijking met mijn vroegere cassetterecorder. Een drive is welliswaar sneller maar vind ik te duur in verhouding met de MDCR. Met een drive zijn de problemen ook de wereld niet uit!!

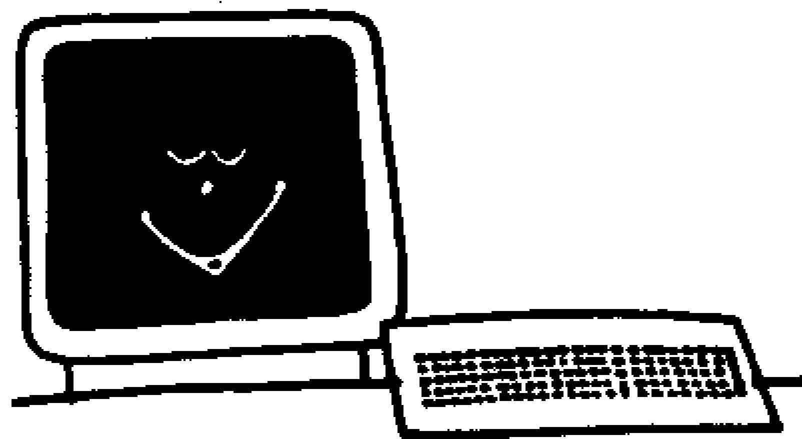
Waar het nu eigenlijk om gaat is het volgende.
Bij "ini" wordt de band "geformatteerd" en dit duurt minimaal twee maal de spoelsnelheid van de MDCR.
De grote truuk met de duif, zonder dat deze dood is, is zeer simpel. Bij "ini" wordt de band eerst geheel teruggespoeld, dan gewist, weer teruggespoeld en vervolgens de Header geplaatst. Indie nu ongeveer 10 sec. nadat het wissen begonnen is heel kort de "esc"-toets ingedrukt wordt, dan begint de band terug te spoelen en de Header wordt geplaatst.
Dit alles duurt dus veel korter dan het hele (nodeloos) wissen van de (lege) band. Tot nu toe heb ik hiermee nog nooit problemen gehad, terwijl de tijdbesparing toch minstens 2*96 sec. is.

Bij het "escapen" simuleer je dus in feite het einde van de band zodat deze terugloopt.

Veel formateer plezier

Jan Swinkels, Heythuysen.

p.s. dit alles zonder geweer...ik bedoel ohne gewahr.



NORM-WIJZIGING ACORN DISK-PROCESSOR.

Het artikel van Nico Stad in het vorige AN beschreef de verplaatsing van de FDC naar #BC48. Hierin zat echter nog een storende fout.... De FDC stond niet alleen op #BC48 maar ook op iedere hogere pagina in het #BXXX blok. Op dit moment zou dat nog niet al te veel problemen opleveren, echter als we de byte's op de andere pagina bij een nieuw clubproject nodig zouden hebben zou dat des te meer problemen opleveren, dus vandaar een wijziging op deze wijziging...

De wijziging:
A6 komt op pen 3 van IC 22 (74LS138)
A5 komt op pen 2 van IC 22 (74LS138)
A4 komt op pen 1 van IC 22 (74LS138)

Pen 9 van IC 14 (74LS42) uit de voet buigen
Pen 20 van IC 18 (2532) uit de voet buigen
Beide pennen doormiddel van een draadje verbinden.

Pen 1, 2 en 3 van IC 11 (74LS02) uit de voet buigen.
A8 komt op pen 1 van IC 11 (74LS02)
A9 komt op pen 2 van IC 11 (74LS02)

Pen 3 van IC 11 (74LS02) aan pen 6 van IC 22 (74LS138) leggen d.m.v. een stukje draad.

Pen 14 van IC 22 (74LS138) uit de voet buigen.
Op de soldeerzijde een draadbrug maken van pen 11 naar pen 14 van IC 22 (74LS138)

De disk-processor staat nu van #BC40 tot #BC50.
Een eprom die reeds geprogrammeerd was voor de plaats #BC48 hoeft niet overgeprogrammeerd te worden, en zal gewoon blijven draaien.

Tabel met de adreslijn aansluitingen op de bus:
A4 --> pen 11
A5 --> pen 10
A6 --> pen 9
A8 --> pen 7
A9 --> pen 28

Bovenstaande aansluitingen zitten op de A-rij van de connector.

De wijziging in de Atom zoals deze besproken is door Nico Stad in het vorige AN blijft gehandhaafd.

Succes.

Naar aanleiding van het vorige artikel in Atom-Nieuws waarin de inleiding van de nieuwe hardware-commissie gegeven werd, dit artikel over de eerste vorderingen van de HWC.
Zaterdag 10 mei j.l. heeft de hardwarecommissie wederom vergaderd, en daarbij enkele spijkers met koppen geslagen. Zo zijn er een aantal nieuwe clubnormeringen bij die, van de "oude" HWC toegevoegd. Deze clubnormeringen zijn terug te vinden op de "centerfold" van dit nummer.
Als eerste, en belangrijkste is er een belangrijke wijziging doorgevoerd binnen het disk-gebeuren van Acorn. Zoals U wellicht zult weten is er al enige tijd sprake van het verplaatsen van de disk-controller van #A00 naar ??. In het voorlaatste AN werd er een wijziging besproken naar het #EXXX blok. Mede door dit artikel dachten wij dat het nu hoog tijd werd om een clubnorm in te stellen. Indien dit niet zou gebeuren kunnen we alleen nog maar raden hoe groot de puinhoop zou worden als iedereen zijn/haar FDC op een andere plaats heeft staan...
De beschrijving van de ombouw kunt U terug vinden in de bijgevoegde handleiding. De basis voor deze wijziging is de wijziging van Nico Stad; echter was daar een foutje ingeslopen; doordat niet alleen #BC48 geselecteerd werd, maar ook alle andere boven dit adres binnen het #BXXX blok liggende blokken (#BDXX,#BEXX,#BFXX) geselecteerd werden. Hierbij dus een definitieve versie...
Waarom we deze methode hebben gekozen? Wel, eigenlijk is dit de meest simpele methode om de FDC te verplaatsen, en bovendien hoort de FDC (I/O) in het #BXXX blok thuis.
De tweede en evenzo belangrijke norm is de plaatsing van de GDOS controller. Deze kaart is verplaatsbaar in de memory-map van de Atom, maar de norm-decodering wordt: PIA:#BC10, FDC:#BC14.
Nogmaals: wij dringen bovenstaande zaken niet aan U op; het is aan U of U de wijzigingen/plaatsingen uitvoert; maar... indien U alles op de clubnorm houdt zal al Uw hardware blijven werken als er nieuwe kaarten uitkomen met een plaatsing in het #BXXX blok.
Natuurlijk wordt er een kleine uitzondering gemaakt voor de Acorn controller. De originele plaats (#0A00) kan gewoon gebruikt blijven worden indien U geen behoefte voelt voor deze wijziging.
De wijziging is tenslotte het meest interessant voor mensen die geheugen hebben van #0000 tot #A000 enz.

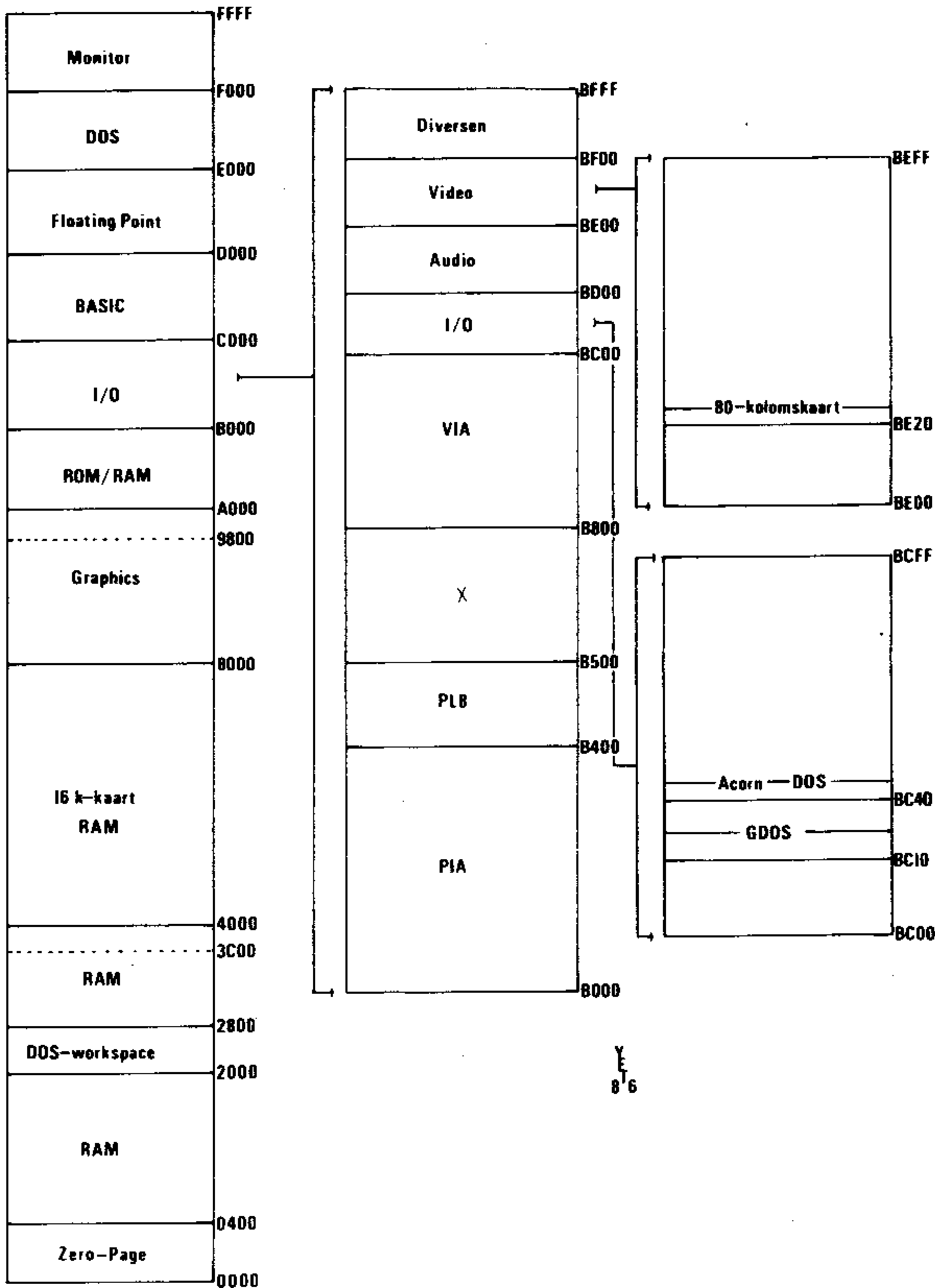
De laatste vorderingen op het gebied van de 80-kolomskaart en GDOS-FDC zijn dat de 80-kolomskaart momenteel in de verkoop is genomen, dit natuurlijk inclusief de onderdelenpaketten.
De print lay-out van de GDOS-FDC is met het uitkomen van dit nummer ook klaar; en dat betekent dat de eerste drie prototypes gemaakt kunnen worden. We hopen op een zo spoedig mogelijke afwikkeling, maar kunnen in deze fase absoluut nog niets beloven over de levertijd van de kaart.

Als laatste onderwerp in dit artikel valt nog te melden dat er tevens een "werknorm" is gekomen van het landelijk bestuur voor wat betreft het

formaat van de diverse kaarten die nog uit moeten komen. Het laat zich niet moeilijk raden dat dit het z.g. eurokaart-formaat is (100 * 160 mm). Wegens technische, en financiele redenen is het echt niet mogelijk om kaarten te produceren die groter zijn dan bovenstaand formaat, tenzij een kaart bijvoorbeeld sowieso niet op de bus aangesloten kan worden. Hierbij valt te denken aan een kaart die bijvoorbeeld slechts enkele I/O aansluitingen heeft. De vormgeving van zo'n kaart zal iedere keer opnieuw bekeken worden. U kunt natuurlijk wel raden dat de HWC na het uitbrengen van de 80-kolomskaart en de GDOS-controller niet bij de pakken neer wil gaan zitten; direct na het verschijnen van deze kaarten zal er gedacht worden aan nieuwe projecten. Deze projecten moeten natuurlijk niet uit de HWC komen maar uit het land. Om deze reden wil ik U vragen te reageren op de vraag wat de HWC allemaal uit zou moeten brengen. Dit kan varieeren tot een klein en simpel printje wat in de praktijk zeer handig is tot een "groot" project. Reacties zien wij gezien de reproduceerbaarheid het liefst schriftelijk. Denk hier s.v.p. eens over na, de HWC is er niet om eigen ideeen uit te bengen!

Namens de hardwarecommissie: Yvo Tuk.




FFFF
Monitor
F000
DOS
E000
Floating Point
D000
BASIC
C000
I/O
B000
ROM/RAM
A000
9800
Graphics
8000
16 k-kaart
RAM
4000
3C00
RAM
2800
DOS-workspace
2000
RAM
0400
Zero-Page
0000
BFFF
Diversen
BF00
Video
BE00
Audio
BD00
I/O
BC00
VIA
B800
X
B500
PLB
B400
PIA
B000
BEFF
80-kolomskaart
BE20
BE00
BCFF
Acorn DOS
BC40
GDOS
BC10
BC00

1. Korte cursus windsurfen

Kort gezegd komt het hele surfambacht neer op drie basis-vaardigheden :

- 1. op de plank kunnen klimmen en erop blijven staan
- 2. starten
- 3. de plank besturen

1.1 op de plank klimmen en blijven staan

Dit moet U uzelf aanleren: gewoon proberen en niet boos worden als familie en/of vriend(inn)en zich kleurenblind lachen om uw herhaaldelijke spartel-partijtjes.

1.2 starten

Als u wilt starten, doet u dat door op de plank te gaan staan met uw rug naar de wind gekeerd.
Het zeil moet dan VOOR u in het water liggen (dus niet aan die kant waar de wind vandaan komt).
Vervolgens trekt u het zeil op door met beide handen aan het touw te trekken dat aan de giek (het handvat van het zeil) bevestigd is.
Zodra dit mogelijk is pakt u de giek met beide handen vast: een hand dicht bij de mast en de andere daar een stukje vandaan (zeg: halverwege de giek)

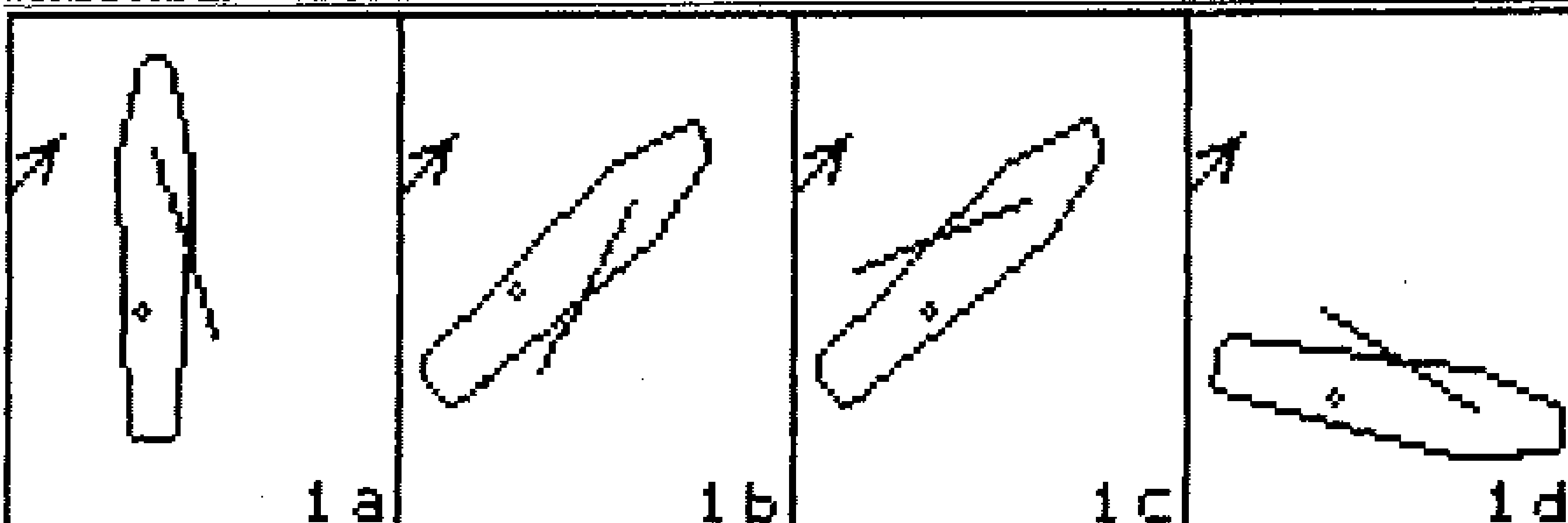
Op dat moment gaat de plank vooruit. Nu moet er echter nog gestuurd worden.

1.3 sturen

De besturing van de surfplank geschiedt in tegenstelling tot vrijwel alle andere vaartuigen uitsluitend met behulp van het zeil.
Door het gehele zeil naar voren te bewegen (dus bewegen in de vaarrichting), draait de plank totdat deze in dezelfde richting vaart als de wind waait.

Door het zeil naar ACHTER te bewegen, draait de plank totdat u pal tegen de wind in vaart.

Als u het verhaal tot nu toe begrepen heeft, is u misschien opgevallen dat we nog niet in alle gewenste richtingen kunnen varen.
Om dit te bereiken moet u overstappen: u beweegt de mast naar voren of naar achter en wacht totdat de plank stopt met draaien: u vaart dan pal tegen wind in of juist met wind mee. Op dat moment stapt u naar de andere kant van het zeil op de plank door eerst voorop de plank te gaan staan (dus VOOR de mast) en daarna weer terug te stappen met het zeil aan de andere kant.
Door nu het zeil naar voren of naar achteren te bewegen draait de plank verder in de door U gekozen richting.
Dit alles is in afbeelding 1 in beeld gebracht.



AFBEELDING 1

1A
Surfer staat links van het zeil, plank vaart min of meer met de wind mee (zie wind-pijl)

1B
Nadat het zeil naar VOREN is bewogen, draait de plank met de klok mee (N.B. in dit voorbeeld !) totdat deze precies in de wind-richting vaart.

1C
De surfer stapt naar de andere kant van het zeil .

1D
Nadat het zeil naar ACHTEREN is getrokken, draait de plank weer TEGEN de wind in, zodat de plank verder draait in klokrichting.

2. het SURFSIM programma

Het surf-simulator-programma maakt het mogelijk om de zojuist besproken windsurf-theorie in 'praktijk' te brengen zonder een nat pak te halen.
U heeft daar dan alleen een ATOM voor nodig (geheugen: #2800..#A000) met een joystick.
Als U het programma voor de eerste keer RUNt dan moet u de vraag 'tabel aanmaken (1=ja)' beantwoorden met '1'.
Op het scherm verschijnt dan een surfplank en een pijl die de windrichting weergeeft, de joystick doet dienst als zeil-mast.
(dus: joystick naar voren = mast naar voren etc.)
U moet nu eerst het zeil optrekken, waarna u het zeil ietwat schuin ten opzichte van de plank moet houden (van bovenaf gezien).

Het programma maakt het mogelijk om te experimenteren met snelheidsveranderingen die optreden bij verschillende standen van het zeil of verschillende vaar-richtingen.

Veel plezier en vergeet niet: echt surfen is toch leuker.

Andre de Bruin
Baljuw 11
2671 HL naaldwijk

DISKCAT LIBRARY SYSTEM.

Zoals Nico Stad reeds in zijn artikel over de 80-kolomskaart beschreef heb ik het volledige diskcat systeem uit de kast gehaald, hier en daar wat opgepoetst, en geschikt gemaakt voor de 80-kolomskaart.
Als U het systeem kent zal het allemaal niet zo moeilijk zijn; U prikt een 80-koomskaart in Uw systeem en fietsen met de hele handel. Indien U het pakket nog niet kent hieronder een beknopte uitleg.
Diskcat is origineel geschreven door Frans le Blanc uit regio Den Haag. Het is een pakket van programma's wat er voor zorgt dat we schijven (geschreven op ACORN-formaat) in een bestand kunnen lezen, en op die manier overzichtelijk op kunnen bergen. Indien U op de eerste schijf in Uw bakje het systeem zet, en het regelmatig bijwerkt, kunt U Uw oude programma's snel terugvinden door gebruik te maken van de find-module. U kunt d.m.v. het pakket ook een gealfabetiseerde uitdraai maken van alle file's die op Uw schijven staan. Bovenstaand geldt ook voor de titels van de schijven.
Hoe moet U het pakket bedienen, en wat is er voor nodig?
Allereerst de benodigdheden op een rij:
P-charme
Geheugen van #2000 tot #9800
Schakelsysteem
DBOX-V1
JOSBOX (Fill)
80-kolomskaart (F.A.C.)

Bediening:
Voor de eerste keer:
Draai het programma STARTDC
indien dit klaar is start U het gehele pakket met *1. (Let wel! het opstartprogramma moet op kant 0 van de schijf staan!!!)
Indien U al een bestand heeft:
Start met *1
De diskdrive zal gaan draaien, en na enige tijd verschijnt er in dubbelhoog gekleurde 40-kolomsmode een menu, van waaruit U de verschillende modules aan kunt roepen.
Uitleg bij de modules:
Find: toets een filename of deel van een filename in, wordt opgezocht.
Catalog: Zet de catalog van iedere gewenste schijf op het scherm.
Print catalog: voert iedere gewenste catalog uit op de printer
Print Index: print alfabetische lijst van alle filename's.
Set new files: invoer van nieuwe of gewijzigde file's. Let op! indien U wlt stoppen met deze module dient U bij de laatste schijf die U invoert niet alleen de RETURN-toets doch ook de CTRL-toets indrukken!!.
Quit: Stopt pakket.

GDOS of Z80 KAART ??

Op de vergadering d.d. 13-9-86 is besloten om het door mij ontworpen Z80 kaartje over te nemen. Door deze kaart aan de ATOM te koppelen ontstaat de mogelijkheid CP/M software te draaien.
Met behulp van de Z80 kaart is het ook mogelijk om ATOM-DOS en GDOS te draaien.
Nu er voor de leden keus is uit twee DOS'sen wil de federatie weten wie de GDOS, en wie de Z80 kaart wil gaan bouwen. Alleen dan kan de federatie de juiste hoeveelheid printplaten bestellen.
Voor veel leden zal het onduidelijk zijn wat de Z80 kaart te bieden heeft en wat niet. Het bestuur heeft mij daarom gevraagd om zo duidelijk mogelijk de verschillen weer te geven tussen de twee systemen wat betreft prijs en mogelijkheden. De leden kunnen dan zien of zij de extra mogelijkheden van Z80 kaart willen hebben en of zij de meerprijs ervoor over hebben.

Allereerst zal ik duidelijk proberen te maken hoe de ATOM met Z80 kaart eruit ziet. In het blokschema hieronder is een ATOM getekend met een extra VIA. M.b.v. deze VIA kan worden gecommuniceerd met de Z80 kaart. De Z80 kaart is voorzien van een disc controller.

```
ATOM met extra 6522 <=====>  Z80 krt <=====>  drives
en 80 kolommenkrt
```

Wanneer men CP/M draait, fungeert de ATOM slechts als intelligente terminal.
Aangezien er nu, weliswaar met een omweg, een disc-controller aan de ATOM hangt is het mogelijk een DOS voor de ATOM te maken. De GDOS en ATOMdos zijn zodanig gewijzigd dat alle disk-IO met een omweg gedaan wordt via de Z80. Alle informatie over tracks sectors etc. wordt overgestuurd naar de Z80. Deze voert de lees of schrijfacties uit, waarna het resultaat teruggestuurd wordt aan de ATOM. Het zal eenieder duidelijk zijn dat door de noodzakelijke extra communicatie, het laden en saven vertraagd wordt. Voor de beide DOS'sen geldt een vertraging van circa 50% tot 100% ten opzichte van de originele DOS'sen.

Ik zal nu iets meer vertellen over de ATOM-DOS implementatie.
Alle commando's uit de originele ROM (TELEC of DECADOS of ATOMDOS) werken gewoon. Uit de DBOX-V1 werken de volgende commando's niet meer: *SECTOR, *FORMAT, *TYPE.
De errorafhandeling is vereenvoudigd: Er wordt geen foutmelding gegeven bij een kapotte schijf. Bij dergelijke calamiteiten loopt de machine vast, zodat men kan raden wat er fout is. Dergelijke fouten komen bij mij circa 1 keer per jaar voor.
Met een handigheidje is het mogelijk om meer directories op 1 schijf te zetten. Heeft men een 2*80 tracks drive (1 Mb), dan kan men 8*31=248 files op 1 schijf kwijt.

Over de GDOS implementatie kan ik kort zijn: Alle kommando's zijn gebleven. Men kan dus ook formatteren vanuit de ATOM. Alle foutmeldingen zijn zoals bij het origineel.
In tegenstelling tot de gewone versie van GDOS, heeft men geen extra hardware nodig voor het softwarematig omschakelen tussen 40 en 80 track floppies.

Wat zijn nu de meest belangrijke verschillen?
- Met de Z80 kaart heeft men naast CP/M ook GDOS en ATOMDOS beschikbaar. Het is echter ook mogelijk met de GDOS hardware de standaard ATOM DOS te draaien. De standaard DOSROM zou hiervoor gewijzigd moeten worden.
- Vanwege de ingewikkelde configuratie is experimenteren met de Z80 kaart lastiger. Door de extra mogelijkheden kan er echter wel meer geprobeerd worden. Men zou b.v. een printerbuffer kunnen programmeren m.b.v. de Z80 kaart.
- GDOS is een stuk goedkoper en laat zich eenvoudiger inbouwen omdat men slechts 1 print hoeft op te bouwen.
- Met Z80 kaart: langzamer ATOM DOS
- Met Z80-kaart is een zwaardere voeding nodig (circa 1 A).

Wat heeft men nodig aan hardware bij een Z80 kaart ?
- Een extra VIA aan de ATOM.
- Een DOS rom in de ATOM op adres E000 (op schakelkaart).
- Nadat men hoofdpijn heeft gekregen van het 64 kolommen grafische scherm wenst men een 80 kolommen kaart.
- Extra kabel met konnektoren tussen VIA en Z80 kaart.
- Disc drive (net als bij GDOS !).

Voor een overzicht wat men nodig heeft voor een originele GDOS verwijs ik naar ATOM NIEUWS 5.2.
Het Z80 kaartje is f75,- tot f100,- duurder dan de GDOS print.
Een 80 kolommen kaart van de klub komt op f140,- (Elektuurkaart is ook bruikbaar)
Een extra VIA inbouwen kost tussen de f15,- en f40,-
De DOS-ROM in de ATOM kost f15,- en kan op een schakelkaart (of piggybacken op FP-rom)

Ik hoop hiermee voldoende duidelijk te zijn geweest. Zijn er desondanks nog dringende vragen, neem dan kontakt op met de hardware commissie of met mij:

Peter Huisken, afd. Twente.

P.S. Voor degenen die precies het verschil in benodigde hardware willen zien, volgt hier de onderdelenlijst van de Z80 kaart:

TTL IC'S	NMOS IC'S	ELKO'S
74LS00	Z-80A CPU	10uF
74LS02 (2*)	Z-80A PIO	68uF
74LS04 (2*)	1793 of MB8877	
74LS06	27128	**WEERSTANDEN**
74LS74	4164 (8*)	
74LS121		220 (5*)
74LS123		1K (5*)
74LS139		2K2
74LS145	**CONDENSATOREN**	10K (4*)
74LS153		39K
74LS157 (2*)	10pF	
74LS193	33pF	**KRISTAL**
74LS195	68pF	
74LS367	10nF	16 Mhz
74LS393	100nF (16*)	

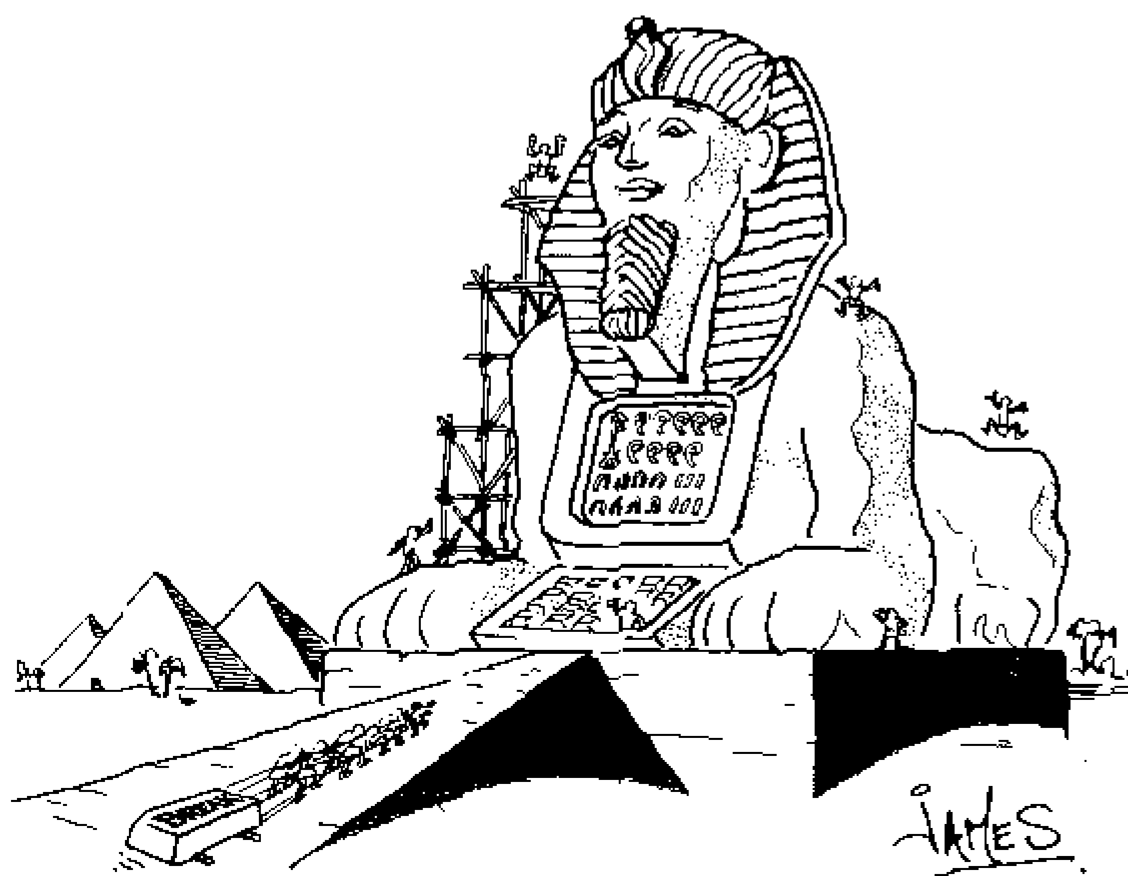
CP/M MET DE Z80-KAART

Door de beschikbaarheid van een 'public domain' vervanger is een CP/M 2.2 licentie niet nodig. Deze vervangende software bestaat uit:
- ZCPR2. Vervangt de zogenaamde CCP.
  Geschreven door R. Conn.
- ZCPR2 utils. Vervangen de CP/M programma's STAT, PIP etc.
  Grotendeels geschreven door R. Conn.
- P2DOS. Vervangt de BDOS.
  Geschreven door H.A.J. ten Brugge.

Naast bovengenoemde software is een op de hardware aangepast BIOS (Basic Input Output System) noodzakelijk. De voor het Z80 kaartje ontwikkelde BIOS heeft de volgende kenmerken:
- Diskformaten instelbaar. Vergelijkbaar met BigBoard II.
- Allerlei soorten 5" en 8" drives aansluitbaar (ook gemixed).
- Variabel formatteren van schijven mogelijk.
- Alle invoer en uitvoer wordt geregeld door de ATOM. Wil men **onder CP/M printen dan kan de printer-kabel gewoon in de ATOM** blijven zitten. Bovendien is alles gebufferd (type-ahead van 60 chars !!!!).
- Eventuele seriële interface in de ATOM functioneert als PUNCH en READER kanaal onder CP/M.

Het terminal programma voor de atom komt in drie versies beschikbaar te weten:
- Klub 80 kolommen kaart (9345)
- Elektuur achtige kaarten (6845)
- Atom grafische scherm, 24*64 (6847)

Binnenkort wordt de komplete handleiding/ beschrijving gepubliceerd. Hierin wordt dan alle software en hardware uitvoerig besproken.



*** CMOS MONITOR ***

INLEIDING

Om een beetje comfortabel met de CMOS-processor te kunnen werken, is een machinetaal monitor eigelijk onmisbaar. Daarom heb ik een nieuwe monitor ontwikkeld die ook met de extra CMOS instructies kan werken. De monitor zelf is in 65C02 assembly geschreven. Daarbij is geen gebruik gemaakt van de extra bit-manipulatie instructies die de Rockwell versie kent. Deze speciale instructies worden overigens wel door de monitor ondersteund.

GEHEUGEN-GEBRUIK

```
SOURCE  #2900..#68F8  of ander gebied, ca. 16K, SALFAA C2.0 nodig
TABLE   #8200..#8B48  of ander gebied, ca. 2.3K, symboltable

OBJECT  #7800..#7FCC  of ander gebied, ca. 2K
ZP      #00A0..#00A6  of andere 7 opeenvolgende ZP locaties
ABS     #012F..#013F  of andere 17 opeenvolgende RAM locaties
BUFFER  #0100..#012E  commando buffer
STACK   #0180..#01FF  wordt in z'n geheel gebruikt

SCREEN  #8000..#81FF  standaard ATOM beeldscherm
OSWRCH  #00DE..#00E7  indirect ZP gebruik door aanroep routine op #FFF4
OSCLI   COS/DOS ZP's  als gebruik gemaakt wordt van een stercommando via #FFF7
```

OPSTARTEN

```
LINK #7800  -->  koude start (reset processor registers en monitor registers)
LINK #7803  -->  warme start (laat bovengenoemde registers onveranderd)
```

Men kan daar eventueel een BASIC-commando voor maken. Bijvoorbeeld:

```
MON: JSR #C4E4
     JSR #7800
     JMP #C2CF
```

COMMANDO'S

Na het opstarten bevindt men zich in de commando-mode van de monitor. Dit is te zien aan de dubbele punt welke als prompt fungeert. De onderstaande lijst geeft een overzicht van de 1-letter commando's welke door de monitor herkend worden. Zowel hoofdletters als kleine letters zijn toegestaan. Een eventuele "afkort-punt" na de commando-letter is facultatief (dus naar keuze te gebruiken of weg te laten). Ook spaties hebben geen invloed. Meervoudige commando's zijn niet toegestaan, dus na de : is steeds maar 1 commando tegelijk mogelijk.
Verder worden alle control-codes herkend (zie AT&P blz 131/132). Dit heeft o.a. als consequentie dat <ESC> of <CTRL+[> een terugkeer naar ATOM BASIC forceert.
Alle adressen en data worden als zijnde hexadecimaal ge-interpreteerd. Hierbij zijn het #-teken en voorafgaande nullen facultatief.
De betekenis van de in het overzicht gebruikte symbolen is als volgt:

```
x,y,z  zijn 16-bits adressen
d      is 8-bits data
r      is een register
<..>   is een keyboard-toets
```

COPY	C x,y,z	kopieert x t/m y naar z (met z<x of z>y). Met read-after-write controle (zie VERIFY).
VERIFY	V x,y,z	vergelijkt x t/m y met z. Onjuiste adressen worden afgedrukt met hun inhoud. Secundaire commando's: <N> = NEXT ONE <SP> = NEXT PAGE <RET> = RETURN
FILL	F x,y,d	vult x t/m y met d. Met read-after-write controle (zie VERIFY).
HEX	H x,y	geeft HEX/ASCII dump van x t/m y
	H x	alleen van x
	H x,	vanaf x en verder
	H ,y	vanaf laatste x-waarde t/m y
	H	alleen van laatste x-waarde
	H ,	vanaf laatste x-waarde en verder secundaire commando's: <N> = NEXT ONE <SP> = NEXT PAGE <RET> = RETURN
DISASS	D x,y	disassembleert van x t/m y alle onder HEX genoemde commando vormen zijn ook hier toegestaan inclusief de secundaire commando's.
MEMORY	M x	zet monitor in memory mode vanaf x
	M	idem vanaf laatste x-waarde geheugen inhoud is te wijzigen door intypen van 2 hex digits of een spatie gevolgd door het gewenste ASCII-karakter. Als nadien de inhoud afwijkt van de ingetypte data, wordt dezelfde locatie nogmaals afgedrukt. secundaire commando's: <N> = NEXT ONE <P> = PREVIOUS ONE <RET> = RETURN
STEP	S x,y	start programma vanaf x in simulator mode tot de program counter (PC) gelijk is aan het display-adres y. Daarna gaat de monitor verder in step mode.
	S x	step mode vanaf x
	S	step mode vanaf huidige waarde van de PC
	S x,	sim mode vanaf x (is gelijk aan: S x,0)
	S ,y	sim mode vanaf huidige PC-waarde tot display-adres y
	S ,	sim mode vanaf huidige PC-waarde (is gelijk aan: S ,0) In step mode worden de registers afgebeeld alsmede een disassembly van de aan de beurt zijnde instructie. Secundaire commando's in step mode: <N> = NEXT ONE (of <SP>) <S> = SKIP ONE <R> = REALTIME EXECUTION (JSR of JMP) <REPT+N> = REALTIME EXECUTION (JSR) <RET> = RETURN Ter indicatie dat de monitor zich in sim mode bevindt, wordt de geheugenplaats welke correspondeert met de rechter bovenhoek van het beeldscherm na elke uitgevoerde instructie gewijzigd. Secundaire commando's in sim mode: <REPT> = REALTIME EXECUTION (JSR) <CTRL+Q> = QUIT (RETURN TO MONITOR COMMAND MODE)

TRACE	T x,y	werkt hetzelfde als STEP maar na het bereiken van het display-adres y gaat de monitor verder in trace mode. De trace mode verschilt van step in de manier waarop de registers en disassembly worden afgebeeld. Dit gebeurt nu in de vorm van 2 kleine beeldscherpjes van elk 5 regels die zelfstandig scrollen. Alle onder STEP genoemde commando vormen zijn ook hier van toepassing.
GO	G x,y	start een programma vanaf x in realtime mode tot de PC gelijk is aan het breakpoint y, waarna met een BREAK-melding naar de monitor teruggekeerd wordt.
	G x	start programma vanaf x
	G	start programma vanaf huidige PC-waarde
	G ,y	start programma vanaf huidige PC-waarde tot breakpoint y. Het opgeven van een x-waarde in het GO commando heeft ALTIJD tot gevolg dat de stackpointer op #FF gezet wordt! Het breakpoint y kan alleen een adres in RAM zijn.
ASIGN REG	r=d	kent de waarde d toe aan een van de processor registers (A,X,Y,S,P) of monitor registers (R,M).
PRINT REG	P	drukt de register set af alsmede een disassembly van de aan de beurt zijnde instructie.
ZERO REG	Z	maakt A,X,Y registers gelijk aan #00, reset het S-register op #FF en initieert het P-register op #34.
STAR COM	*	voert het betreffende ster-commando uit (COS/DOS).
QUIT	Q	verlaat monitor en keert terug naar aanroepende programma

REALTIME SUBROUTINE FACILITEITEN

Er zijn 3 mogelijkheden ingebouwd om subroutines in realtime mode uit te laten voeren tijdens STEP of TRACE mode, namelijk door middel van:

- het R-register
- de REPEAT-toets
- de R-toets

Het R-register is het Realtime page register. Subroutine calls waarvan het hoge adres-byte groter is dan of gelijk is aan de waarde van het R-reg zullen automatisch in realtime mode uitgevoerd worden (op volle processor snelheid dus). Dit gebeurt niet alleen in step/trace mode maar ook in simulator mode! De default inhoud van het R-reg is #F8 zodat timingsgevoelige routines zoals #FE66 e.d. automatisch op volle snelheid uitgevoerd worden. De invloed van het R-reg kan uitgeschakeld worden door er #00 in te zetten.

Als men tijdens sim/step/trace mode de REPEAT-toets ingedrukt houdt, wordt elke JSR instructie die de monitor tegenkomt in realtime mode uitgevoerd. Door wel of niet indrukken van de REPT-toets kan men dus naar keuze een subroutine in 1 klap uit laten voeren of stap voor stap doorlopen. Zodoende kan men reeds geteste subroutines snel passeren.

De bovengenoemde faciliteiten gelden alleen voor JSR instructies. Het komt echter nog al eens voor dat een subroutine afgesloten wordt met bv. JMP #FFF4 i.p.v. JSR #FFF4;RTS. Om ook in dat soort situaties de routine in realtime te kunnen passeren is een 3e mogelijkheid aanwezig. Als men bij de betreffende

JMP op de R-toets drukt dan zet de monitor deze JMP instructie intern om in een JSR + RTS en voert de JSR in realtime mode uit. De R-toets kan ook bij JSR instructies gebruikt worden i.p.v. <REPT+N>. De routine op #FFF4 kan in step/trace mode niet stap voor stap doorlopen worden i.v.m. "shared" zp-adressen!

Als men een subroutine in realtime mode uit wil laten voeren dan moet dit wel een echte subroutine zijn. De tekstprint routine op #F7D1 is bijvoorbeeld geen echte subroutine! Als men zo'n "subroutine" tegenkomt en men drukt op de R-toets dan verliest men de controle over het programma. De processor neemt dan de touwjes in handen en voert de rest van het programma in realtime uit. Bij zulke "subroutines" zit er derhalve niets anders op dan ze stap voor stap te doorlopen. Als het de tekstprint routine betreft kan men dit eventueel omzeilen door step/trace mode even te onderbreken (RETURN-toets) en vervolgens de zaak voort te zetten met: S ,#F7E9. Het is vanwege deze printroutine dat #FB als default waarde voor het R-register gekozen is.

ONDERBREKEN EN VOORTZETTEN

Een met G.x,y gestart programma kan na BREAK AT YYYY voortgezet worden met:

- STEP of TRACE (alle mogelijke vormen)
- GO G of G,y (dus vanaf huidige PC-waarde)
- GO G.x of G.x,y (mits de stackpointer op #FF staat)

Een met STEP/TRACE gestart programma kan na onderbreking voortgezet worden met:

- STEP of TRACE (alle mogelijke vormen)
- GO (alle mogelijke vormen mits de stackpointer op #FF staat)

Dit verschil tussen STEP/TRACE en GO heeft te maken met de manier waarop zij aan het eind van een programma terugkeren naar de monitor.
Een te debuggen programma moet met een RTS instructie worden afgesloten. Deze RTS geeft het logisch einde van het programma aan. (zoals END in BASIC).
Het GO commando keert bij het bereiken van deze RTS naar de monitor terug via een terugkeer-adres op de stack (op de locaties #1FE en #1FF). Dit werkt dus precies hetzelfde als bij LINK vanuit BASIC.
Bij het STEP/TRACE commando daarentegen wordt naar de monitor teruggekeerd als bij het uitvoeren van een RTS instructie de stackpointer op #FF staat.
Dankzij deze constructie is het mogelijk om in sim/step/trace mode programma's te verwerken die de stackpointer resetten (LDX @#FF;TXS). Bijvoorbeeld:

```
>LINK #7800                     start monitor
:S.C2CF,0                       start BASIC-interpreter en simuleer until 0
>PRINT "BELL",$7,'              type deze opdracht in en let daarbij
BELL                            op de rechter bovenhoek van het beeldscherm
>                               met CTRL+Q terug naar monitor
:P                              print registers
 PC   A  X  Y  S NV BDIZC  R M
CD1B 1B 00 00 FD 00110101 FB 1
CD1B C9 7F     CMP @#7F
:Q                              met QUIT terug naar ATOM BASIC
>
```

Op dezelfde manier is het zelfs mogelijk om een BASIC programma te RUNnen terwijl de monitor in simulator mode werkt. Zorg ervoor dat zo'n BASIC-programma niet te lang is, want bij een ERROR of END wordt steeds de TOP opgezocht en goedgezet hetgeen nogal wat tijd kost.
De invloed van de REPT-toets laat zich goed demonstreren bij het LIST commando.

Uit diverse experimenten is gebleken dat de simulator gemiddeld ongeveer een factor 3500 trager werkt dan de echte CPU. Dit betekent dat een klus die de CPU normaal in 1 seconde afwerkt in sim mode bijna 1 uur in beslag neemt! Als er realtime subroutines tussen zitten zal deze verhouding uiteraard anders zijn. Die traagheid wordt overigens voor het grootste deel veroorzaakt door de zeer inefficiente manier waarop de search routine uit de disassembler de benodigde instructie lengte en adresserings mode bepaalt.

Voor het debuggen van een zelf ontworpen BASIC commando (dat dus alleen in direct mode werkt) is het aan te raden dit niet aan het begin van de regel in te typen maar ergens halverwege. Op deze wijze kan men het debug-proces eventueel onderbreken en gebruik maken van monitor commando's zonder dat deze het te debuggen BASIC commando overschrijven. Bijvoorbeeld:

```
>LINK #7800                          start monitor
:T.C2CF,YYYY                         YYYY is start-adres van te testen commando
>              LMOVE 10,40,75        ca. 13 spaties vooraf
```

BREAKPOINT FACILITEIT

Voor de breakpoint faciliteit wordt gebruik gemaakt van een JSR instructie. Het voordeel t.o.v. de BRK instructie is dat nu ook programma's gedebugged kunnen worden die zelf gebruik maken van de BRK instructie. Verder levert het gebruik van de JSR instructie een uniek breakpoint op omdat er altijd maar 1 aanwezig is, wat overigens ook als een nadeel opgevat kan worden. Ook een nadeel is dat er nu 3 bytes ge-installeerd moeten worden tegenover 1 byte bij BRK. Dit kan in sommige gevallen voor problemen zorgen. Bijvoorbeeld:

```
3000 A2 90     LDX @#90          3000 A2 90     LDX @#90
3002 BD 00 31  LDA #3100,X       3002 BD 00 31  LDA #3100,X
3005 C9 0D     CMP @#0D          3005 C9 0D     CMP @#0D
3007 D0 02     BNE #300B         3007 D0 02     BNE #300B
3009 E6 80     INC #80           3009 20 xx xx  JSR MONBRK
300B CA        DEX
300C D0 F4     BNE #3002         300C D0 F4     BNE #3002
300E 60        RTS               300E 60        RTS
```

Stel dat we bij het linker onzin programmaatje een break willen genereren als de teller ?#80 wordt opgehoogd. Na G.3000,3009 verandert de monitor het programmaatje (tijdelijk) zoals dat rechts is afgebeeld. Omdat ook de DEX instructie door het breakpoint overschreven wordt gaat het hier mis. Men moet dus een beetje opletten waar men een breakpoint zet. Een 3-bytes instructie verdient hierbij de voorkeur want dan kan er niets mis gaan.

Na een GO commando met breakpoint kan op 2 manieren naar de monitor teruggekeerd worden. Ofwel door het bereiken van het logisch einde ofwel door het bereiken van het breakpoint. In het laatste geval wordt de melding BREAK AT YYYY gegeven. In beide gevallen wordt de oorspronkelijke data op y, y+1 en y+2 teruggezet. Als om welke reden dan ook niet meer naar de monitor teruggekeerd wordt, blijft het breakpoint bestaan hetgeen betekent dat de oorspronkelijke data als verloren beschouwd moet worden.

Uit het bovenstaande blijkt dat bij het gebruik van een breakpoint de nodige voorzichtigheid geboden is. Maak er daarom alleen gebruik van als dat echt nodig is (bijv. als het gaat om snelheid of timing).
Voor de meeste problemen is het STEP/TRACE commando met display-point y toereikend. Een display-point tast het te debuggen programma niet aan en kan derhalve ook op een ROM-adres gezet worden.

MODE

Het M-register is een monitor register dat fungeert als Mode-flag. Hiermee wordt bepaald of de monitor in NMOS mode (0) of in CMOS mode (1) werkt. Dit is van invloed op de werking van DISAS, STEP en TRACE. De default waarde is 1.

FOUTMELDINGEN

```
COMMAND ERROR  onbekend commando;
               onbekend register;
               parameter te weinig in COPY, VERIFY of FILL.

STACK ERROR AT XXXX  inhoud stackpointer lager dan #C3;
                     niet toegestane voortzetting met GO.

OPCODE ERROR AT XXXX  onbekende opcode.
```

FIXED ENTRY'S

Aan het begin van de monitor zijn enkele begin-adressen samengebracht in een tabel zodat een en ander onafhankelijk van de versie vast ligt.
De disass routine kan bijvoorbeeld gebruikt worden om de BASIC commando's DIS en CDIS te realiseren. Deze routine is ook geschikt voor gebruik in een 2-koloms disassembler. Er worden altijd 30 karakters afgedrukt zonder CRLF aan het eind. De search routine kan bijv. gebruikt worden voor een relocator.

```
MON+0  JMP coldstart
MON+3  JMP warmstart
MON+6  JMP xdisass
MON+9  JMP xsearch
MON+12 .BYT zp'base
MON+13 .WRD abs'base
```

```
xdisass:  in:  zp,zp+1 = adres     out:  disassembly
               AREG = mode               zp+6 = length-1

xsearch:  in:  zp,zp+1 = adres     out:  AREG en zp+6 = length-1
               AREG = mode               abs+6 = addressing mode
                                         XREG = mnemonic index
```

NMOS VERSIE

Omdat de meeste clubleden waarschijnlijk nog met de oude NMOS 6502 processor werken heb ik tevens een NMOS versie gemaakt. In vergelijking tot de bekende MONITOR.RH biedt de nieuwe monitor meer en krachtiger mogelijkheden. Een ander voordeel is het geringer zero-page gebruik. Daar staat echter tegenover dat de object code meer geheugenruimte in beslag neemt (ca. 1.75 K nodig).

SLOTWOORD

De monitor maakt gebruik van de specifieke mogelijkheden van het standaard beeldscherm. Dit betekent dat deze monitor niet zonder meer op systemen met een 80 kolomskaart kan werken. Wellicht dat iemand met een 80 kolomskaart zich geroepen voelt om de nodige aanpassingen aan te brengen voor een CMON80 en/of NMON80 versie. Ik heb daartoe behalve de source van CMON32 en NMON32 ook enige technische informatie aan de redactie toegestuurd.
Hetzelfde geldt eigenlijk voor het toetsenbord, maar een ieder is vrij om de zaak naar eigen (systeem) wensen aan te passen.

```
10   REM ----------
20   REM CMON32 4.2
30   REM ----------
40
50   ROM1;\ SALFAA C2.0
60
70   C=#7800;\ CODE (CA 2K)
80   M=#7800;\ RAM
90   Z=#A0   ;\ ZP (7)
100  A=#12F  ;\ ABS (17)
110  N=23    ;\ TABLEN
120
130  DIM B(N*3)
140
150  PASS0;GOSUB a
160  PASS1;GOSUB a
170
180  PRINT "MONBEG"&monbeg,'
190  PRINT "MONEND"&monend,'
200  PRINT "DISEND"&disend,'
210  PRINT "SIMEND"&simend,''
220
230  INPUT "DIS TEST Y/N "$B
240  IF C=M IF $B="Y" GOSUB t
250  END
260
270 aASM-BEGIN
280
290             .OPTION :11000000
300             .CODE C
310             .RAM M
320
330              \ ------------
340              \ DECLARATIONS
350              \ ------------
360
370 :ack    = #06
380 :bell   = #07
390 :lf     = #0A
400 :cls    = #0C
410 :cr     = #0D
420 :home   = #1E
430 :eot    = #EA
440 :dept   = #40
450 :skip1  = #24
460 :skip2  = #2C
470
480 :curpos = #E0
490 :cursor = #E1
500 :pagmod = #E6
510
520 :ptr1 = Z+0
530 :ptr2 = Z+2
540 :ptr3 = Z+4
550 :len  = Z+6
560
570 :pc   = ptr1
580 :tmp1 = ptr3
590 :tmp2 = ptr3+1
600 :tmp3 = len
610
620 :cmd  = A+0
630 :byt1 = A+1
640 :byt2 = A+3
650 :byt3 = A+3
660 :jmp1 = A+4
670 :jmp2 = A+7
680 :preg = A+10
690 :sreg = A+11
700 :yreg = A+12
710 :xreg = A+13
720 :areg = A+14
730 :rreg = A+15
740 :mode = A+16
750
760 :tmp4   = jmp1
770 :page   = jmp1+1
780 :adm    = jmp1+2
790 :pcsav  = jmp2+1
800
810 :cmdbuf  = #0100
820 :stack   = #0100
830 :stack1  = #0200-dept
840 :stack2  = #0200-2*dept
850
860 :screen    = #8000
870 :portb     = #B001
880 :portc     = #B002
890 :getcmd    = #CD0F
900 :mnetabl   = #F154
910 :mnetabr   = #F194
920 :admtab1   = #F1D5
930 :admtab2   = #F1E4
940 :opctab1   = #F1F3
950 :lentab    = #F202
960 :opctab2   = #F210
970 :admgrp    = #F250
980 :decadr    = #F668
990 :incadr    = #F671
1000 :pradrsp  = #F7F1
1010 :prbytsp  = #F7FA
1020 :prspace  = #F7FD
1030 :prbyt    = #F802
1040 :prnibl   = #F80B
1050 :prbit    = #F813
1060 :getbufch = #F875
1070 :asctohex = #F87E
1080 :getadr   = #F893
1090 :incadrcmp= #FA08
1100 :cmpadr   = #FA0E
1110 :invert   = #FD44
1120 :frame    = #FE66
1130 :scankb   = #FE71
1140 :rdchar   = #FFE3
1150 :rdecho   = #FFE6
1160 :prascii  = #FFE9
1170 :prcrlf   = #FFED
1180 :prchar   = #FFF4
1190 :starcmd  = #FFF7
1200 :intvec   = #FFFE
1210
1220                  \ -------
1230                  \ MONITOR
1240                  \ -------
1250
1260 :monbeg
```

```
1270
1280              \ fixed entry's:
1290
1300              JMP start \ cold
1310              JMP nxtcmd \ warm
1320              JMP xdisass'
1330              JMP xsearch
1340             .BYT Z \ zp
1350             .WRD A \ abs
1360
1370 :start       JSR zero
1390              SEC \ cmos
1400              ROL mode
1410              LDA @#F8
1420              STA rreg
1430
1440 :nxtcmd      CLD
1460              LDA #52
1470              PHA
1480              LDX #07
1490              LDA @":"
1500              JSR getcmd
1510              STX #07
1520              TAX \ A=0
1530              PLA
1540              STA #52
1550              STZ page
1560
1570              LDY @#FF
1580 :nxtbufch    JSR getbufch
1600              CMP @cr
1610              BEQ nxtcmd
1620              CMP @#21
1630              BCC nxtbufch
1640              CMP @"*"
1650              BNE no'star
1660
1670 :nxtmov      INY
1690              LDA cmdbuf,Y
1700              STA cmdbuf,X
1710              INX
1720              CMP @cr
1730              BNE nxtmov
1740              JSR starcmd
1750              BRA nxtcmd
1760
1770 :no'star     AND @#DF
1790              STA cmd
1800              TAX
1810              JSR getbufch
1820              CMP @","
1830              BEQ cmd'srch
1840              CMP @"="
1850              BEQ cmd'srch+1
1860              DEY
1870 :cmd'srch    TXA
1890              LDX @(adrtab-cmdtab-1)&#FF
1900              CMP cmdtab,X
1910              BEQ cmd'fnd
1920              DEX
1930              BPL cmd'srch+3
1940
1950 :cmd'err     JSR pr'com'err
1970 :nxtcmd'     BRA nxtcmd
1990
2000 :cmd'fnd     PHX
2020              CMP @"="
2030              BEQ getadr3
2040              JSR getadr2
2050              STA tmp4
2060              CPX @","
2070              BNE P+5
2080              JSR setpag
2090              ASL A
2100              BEQ default
2110              LDA ptr2
2120              LDX ptr2+1
2130              JSR to'ptr1
2140 :default     JSR getadr2
2160              TAX
2170              BEQ getadr3
2180              ASL cmd
2190 :getadr3     LDX @ptr3
2210              JSR getadr'
2220              PLY
2230              CPY @5
2240              BCS skp'inc
2250              CPY @3
2260              BCS P+5
2270              TAX
2280              BEQ cmd'err
2290              LDX @(ptr2+#A6)&#FF
2300              JSR incadr
2310 :skp'inc     LDA pagmod
2330              PHA
2340              ORA @#80 \ off
2350              STA pagmod
2360              JSR exec'cmd
2370              PLA
2380              STA pagmod
2390              BRA nxtcmd'
2400
2410 :exec'cmd    TYA
2430              ASL A
2440              TAX
2450              JMP (adrtab,X)
2460
2470 :getadr2     LDX @ptr2
2490 :getadr'     JSR getbufch
2510              CMP @"#"
2520              BEQ getadr'
2530              JSR getadr
2540              LDX cmdbuf,Y
2550              CPX @cr
2560              BNE P+3
2570              DEY
2580              RTS
2590
2600 :fill        LDA ptr3
2620              STA tmp3
2630              LDA ptr1
2640              LDX ptr1+1
2650              STA ptr3
2660              STX ptr3+1
2670              BRA next
2680
```

```
2690 :copy
2700 :verify    ASL cmd
2720 :next      LDA (ptr1)
2740            BIT cmd
2750            BVS compare
2760            BPL store
2770            LDA tmp3
2780 :store     STA (ptr3)
2800 :compare   CMP (ptr3)
2820            BEQ cmp'ok
2830            LDX @ptr3
2840            JSR pradrsp
2850            LDA (ptr3)
2860            JSR prhexasc
2870            JSR control
2880            BCS ret0
2890 :cmp'ok    JSR incptr31
2910            BCC next
2920 :ret0      RTS
2940
2950 :nxtdis    JSR control'
2970            BCS ret1
2980 :dis       JSR xdis'
3000            LDY len
3010            INY
3020            JSR updptr1
3030            BCC nxtdis
3040 :ret1      RTS
3060
3070 :nxthex    JSR control
3090            BCS ret1
3100 :hex       LDY @3
3120            STY len
3130            JSR pradrbyts
3140            LDY @0
3150 :nxtasc    JSR prspace
3170            LDA (ptr1),Y
3180            JSR pr'asc
3190            INY
3200            CPY @4
3210            BNE nxtasc
3220            JSR updptr1
3230            BCC nxthex
3240            BRA crlf
3250
3260 :prhexasc  PHA
3280            JSR prbytsp
3290            PLA
3300 :pr'asc    CMP @" "
3320            BCC P+6
3330            CMP @#7F
3340            BCC P+4
3350            LDA @"."
3360            JMP prchar
3370
3380 :no'ram    LDA @bell
3400            JSR prchar
3410 :nxtmem    JSR prcrlf
3430 :memory    JSR pr'ptr1
3450            LDA (ptr1)
3460            JSR prhexasc
3470            JSR prspace
3480 :rdmem     JSR rwbyte
3500            BCS no'hex
3510            ORA tmp1
3520            STA (ptr1)
3530            CMP (ptr1)
3540            BNE no'ram
3550 :nextmem   JSR incptr1
3570            BRA nxtmem
3580 :no'hex    AND @#DF
3600            CMP @"N"
3610            BEQ nextmem
3620            CMP @"P"
3630            BNE no'prev
3640            LDX @(ptr1+#A6)&#FF
3650            JSR decadr
3660            BRA nxtmem
3670 :no'prev   CMP @cr
3690            BNE rdmem
3700 :crlf      JMP prcrlf
3720
3730 :rwbyte    JSR rwnibl
3750            BCS tst'sp
3760            ASL A
3770            ASL A
3780            ASL A
3790            ASL A
3800            STA tmp1
3810
3820 :rwnibl    JSR rdnibl
3840            BCS ret2
3850            PHA
3860            JSR prnibl
3870            PLA
3880            CLC
3890 :ret2      RTS
3910
3920 :tst'sp    CMP @" "
3940            SEC
3950            BNE ret2
3960            JSR prchar
3970            JSR rdecho
3980            STA tmp1
3990            CLC
4000            RTS
4010
4020 :rdnibl    JSR rdchar
4040            CMP @"G"
4050            BCS ret2
4060            CMP @"A"
4070            BCS P+6
4080            CMP @CH"9"+1
4090            BCS ret2
4100            JMP asctohex
4110
4120 :step      JSR prepare
4140            BRA tst'cmd
4150
4160 :trace     JSR prepare
4180            BPL clr'scr
4190
4200 :go'on1    LDA portb
4220            AND @:01010000 \ CTRL+Q
4230            BEQ return'
4240            LDX @ptr1
```

```
4250              JSR cmpadr
4260              STA screen+#1F
4270              BNE go'on2
4280              LSR cmd
4290              JSR ack'crlf
4300              LDA cmd
4310              CMP @"T"
4320              BNE tst'cmd
4330 :clr'scr     LDA @cls
4350              JSR prchar
4360              JSR coff
4370              BRA tst'cmd
4380
4390 :go'on2      LDA mode
4410              AND @:00111111
4420              STA mode
4430              JSR xsearch
4440 :go'on3      LDA sreg
4460              CMP @#0103-dept
4470              BCC stack'err
4480              LDX adm
4490              CPX @#0F
4500              BEQ opcode'err
4510              JSR sim
4520 :tst'cmd     LDA cmd
4540              BMI go'on1
4550 :return'     BEQ return
4570              CMP @"T"
4580              BEQ nxt'trace
4590
4600 :nxtstep     JSR prcrlf
4620              JSR printreg+3
4630 :nxtstep'    STZ page
4650              LDA @:11000000
4660              TRB mode
4670              JSR control'
4680              BCC go'on3
4690              BRA return
4700
4710 :nxt'trace   JSR upd'screen
4730              BRA nxtstep'
4740
4750 :stack'err   JSR ack'crlf
4770             .ASC "STACK "
4780              BRA pr'err
4790
4800 :opcode'err  JSR ack'crlf
4820             .ASC "OPCODE ",eot
4830
4840 :pr'err      JSR print
4860             .ASC bell,"ERROR",eot
4870
4880 :pr'at       JSR print
4900             .ASC " AT ",eot
4910              JSR pr'ptr1
4920
4930 :return      JSR con
4950              JSR ack'crlf
4960              LDA ptr1
4970              LDX ptr1+1
4980              STA pcsav
4990              STX pcsav+1
5000
5010 :swap        CLC
5030              LDX @dept-1
5040              LDY stack1,X
5050              LDA stack2,X
5060              STA stack1,X
5070              TYA
5080              STA stack2,X
5090              DEX
5100              BPL swap+3
5110              TSX
5120              TXA
5130              BCC P+4
5140              SBC @2*dept+1
5150              ADC @dept
5160              TAX
5170              TXS
5180              RTS
5190
5200 :go          JSR prepare
5220              LDY @#20 \ JSR
5230              LDX sreg
5240              INX
5250              BEQ stack'ok
5260              LDX tmp4
5270              BEQ continue
5280              JSR reset'sreg
5290              BRA stack'ok
5300 :continue    JSR tst'stack
5320              BNE stack'err
5330              LDY @#4C \ JMP
5340 :stack'ok    STY byt1
5360              LDY @0
5370              BVS no'brkpnt
5380 :nxtpush     LDA (ptr2),Y
5400              PHA
5410              LDA brkpnt,Y
5420              STA (ptr2),Y
5430              INY
5440              CPY @3
5450              BNE nxtpush
5460              DEY
5470 :no'brkpnt   JSR init'
5490              LDX @realtime%256
5500              LDY @realtime/256
5510              JSR execute'
5520              BVS return
5530              ASL A
5540              LDY @2
5550 :nxtpull     PLA
5570              STA (ptr2),Y
5580              DEY
5590              BPL nxtpull
5600              BCC return
5610              JSR pull'pc
5620              LDA @#FE \ -2
5630              JSR upd'pc'
5640              JSR ack'crlf
5650             .ASC "BREAK",eot
5660              JMP pr'at
5670
5680 :brkpnt      JSR monbrk
5700
5710 :realtime    JMP (pc)
```

```
5730
5740 :prepare     SEC
5760              JSR swap+1
5770              LDA tmp4
5780              BNE ret3
5790 :ldpc        LDA pcsav
5810              LDX pcsav+1
5820 :to'ptr1     STA ptr1
5840              STX ptr1+1
5850 :ret3        BIT cmd
5870              RTS
5880
5890 :printreg    JSR ldpc
5910              JSR prheader'
5920 :pr'val      JSR pr'values
5940 :xdis'       LDA mode
5960              JSR xdisass'
5970              BRA prcrlf'
5980
5990 :upd'screen JSR prheader
6010              LDY @#80
6020              JSR frame
6030 :nxtscroll   LDA screen-#20,Y
6050              STA screen-#40,Y
6060              LDA screen+#A0,Y
6070              STA screen+#80,Y
6080              INY
6090              BNE nxtscroll
6100              JSR prlf5
6110              JSR pr'values
6120              JSR prlf5
6130              BRA xdis'
6140
6150 :prlf5       LDX @5
6170              LDA @lf
6180              JSR prchar
6190              DEX
6200              BNE P-4
6210              RTS
6220
6230 :prheader    LDA @home
6250              JSR prchar
6260 :prheader'   JSR print
6280             .ASC " PC   A  X  Y  S"
6290             .ASC " NV BDIZC  R M"
6300 :prcrlf'     BRA prcrlf''
6320
6330 :pr'values   JSR pr'ptr1
6350              LDX @4
6360 :nxtreg      LDA preg,X
6380              JSR prbytsp
6390              DEX
6400              BNE nxtreg
6410              LDA preg
6420              LDY @8
6430 :nxtbin      ASL A
6450              PHA
6460              TXA
6470              JSR prbit
6480              PLA
6490              DEY
6500              BNE nxtbin
6510              JSR prspace
6520              LDA rreg
6530              JSR prbytsp
6540              LDA mode
6550              AND @#01
6560              JSR prnibl
6570 :prcrlf''    JMP prcrlf
6590
6600 :zero        STZ areg
6620              STZ xreg
6630              STZ yreg
6640              LDA @:00110100
6650              STA preg
6660 :reset'sreg LDA @#FF
6680              STA sreg
6690              RTS
6700
6710 :asign       LDX @(cmdtab-regtab-1)&#FF
6730              LDA cmd
6740 :nxt'asign   CMP regtab,X
6760              BEQ asign'fnd
6770              DEX
6780              BPL nxt'asign
6790
6800 :pr'com'err JSR print
6820             .ASC bell,"COMMAND ERROR"
6830              BRA prcrlf''
6840
6850 :asign'fnd   LDA ptr3
6870              STA preg,X
6880              RTS
6890
6900 :quit        PLA
6920              PLA
6930              PLA
6940              STA pagmod
6950              RTS
6960
6970 :control     JSR prcrlf
6990 :control'    JSR scankb
7010              CPY @cr
7020              BEQ ret4
7030              LDX page
7040              BNE setpag+2
7050              CPY @#2E \ N
7060              BEQ ret4-1
7070              TYA \ SP
7080              BEQ space'bar
7090              LDA @:01000000
7100              CPY @#32 \ R
7110              BEQ upd'mode
7120              CPY @#33 \ S
7130              BNE control'
7140              ASL A
7150 :upd'mode    TSB mode
7170              CLC
7180              RTS
7190 :space'bar   JSR prcrlf
7210 :setpag      LDX @15
7230              DEX
7240              STX page
7250              CLC
7260 :ret4        RTS
7280
```

```
7290 :pr'ptr1     LDX @ptr1
7310              JMP pradrsp
7320
7330 :incptr31    LDX @(ptr3+#A6)&#FF
7350              JSR incadr
7360 :incptr1     LDY @1
7380 :updptr1     LDX @ptr1
7400              JSR incadrcmp
7410              BEQ ret5
7420              DEY
7430              BNE updptr1+2
7440              CLC
7450 :ret5        RTS
7470
7480 :ack'crlf    JSR print
7500             .ASC ack,cr,eot
7510 :print       PLA
7530              STA ptr3
7540              PLA
7550              STA ptr3+1
7560 :nxtpr       INC ptr3
7580              BNE P+4
7590              INC ptr3+1
7600              LDA (ptr3)
7610              BMI endpr
7620              JSR prascii
7630              BRA nxtpr
7640 :endpr       JMP (ptr3)
7660
7670 :con         LDA @#80
7690             .BYT skip2
7700 :coff        LDA @0
7720              CMP cursor
7730              BEQ ret5
7740              STA cursor
7750              LDY curpos
7760              JMP invert
7770
7780 :regtab     .ASC "PSYXARM"
7800
7810 :cmdtab     .ASC "CVFHDSTGM=PZQ"
7830
7840 :adrtab     .WRD copy
7860             .WRD verify
7870             .WRD fill
7880             .WRD hex
7890             .WRD dis
7900             .WRD step
7910             .WRD trace
7920             .WRD go
7930             .WRD memory
7940             .WRD asign
7950             .WRD printreg
7960             .WRD zero
7970             .WRD quit
7980
7990 :monend
8000
8010              \ -----------
8020              \ DISASSEMBLER
8030              \ -----------
8040
8050 :xdisass'    JSR xdisass
8070
8080 :fetch       LDY @2
8100              LDA (ptr1),Y
8110              STA byt1,Y
8120              DEY
8130              BPL fetch+2
8140              RTS
8150
8160 :xdisass     JSR xsearch
8180              PHX
8190              JSR pradrbyts
8200              LDA spctab,Y
8210              TAY
8220              JSR pryspc
8230
8240              PLX
8250              BPL skipnew
8260              LDA mnemr-#80,X
8270              LDY meml-#80,X
8280              BRA skipold
8290 :skipnew     LDA mnetabr,X
8310              LDY mnetabl,X
8320 :skipold     STA tmp1
8340              STY tmp2
8350              LDY @3
8360 :nxtchar     LDX @5
8380              LDA @0
8390 :nxtrol      ROL tmp1
8410              ROL tmp2
8420              ROL A
8430              DEX
8440              BNE nxtrol
8450              ADC @#3F
8460              JSR prchar
8470              DEY
8480              BNE nxtchar
8490              JSR prspace
8500
8510              LDY @11
8520              LDX adm
8530              BPL normal
8540
8550              LDA (ptr1)
8560              AND @#70
8570              LSR A
8580              LSR A
8590              LSR A
8600              LSR A
8610              JSR prnibl
8620              JSR print
8630             .ASC ",#"
8640              LDA byt2
8650              JSR prbyt
8660              LDY @6
8670              DEX
8680              BMI pryspc
8690              JSR print
8700             .ASC ","
8710              LDA byt3
8720              STA byt2
8730              LDX @2
8740              DEY
8750
```

```
8760 :normal     STY tmp1
8780             LDY @7
8790             LDA admformat,X
8800             STA tmp2
8810 :nxtbit     CPY @5
8830             BNE nxtasl
8840             TXA
8850             BNE skiprel
8860
8870             LDA byt2
8880             BPL P+3
8890             DEX
8900             ADC ptr1
8910             PHA
8920             TXA
8930             ADC ptr1+1
8940             TAX
8950             PLA
8960             CLC
8970             ADC len
8980             STA byt2
8990             TXA
9000             ADC @0
9010             STA byt3
9020             LDX @2
9030             BRA prhex
9040
9050 :skiprel    LDX len
9070             BEQ nxtasl
9080 :prhex      LDA @"#"
9100             JSR prchar
9110             DEC tmp1
9120 :nxtbyt'    LDA byt1,X
9140             JSR prbyt
9150             DEC tmp1
9160             DEC tmp1
9170             DEX
9180             BNE nxtbyt'
9190 :nxtasl     ASL tmp2
9210             BCS skipchar
9220             LDA admchar,Y
9230             JSR prchar
9240             DEC tmp1
9250 :skipchar   DEX
9270             DEY
9280             BPL nxtbit
9290             LDY tmp1
9300
9310 :pryspc     INY
9330             DEY
9340             BEQ ret6
9350             JSR prspace
9360             BRA pryspc+1
9370
9380 :pradrbyts  JSR pr'ptr1
9400             LDY @#FF
9410 :nxtbyt     INY
9430             LDA (ptr1),Y
9440             JSR prbytsp
9450             CPY len
9460             BNE nxtbyt
9470 :ret6       RTS
9490
```

```
9500 :xsearch    LSR A
9520             JSR fetch
9530             BCC oldsearch
9540             TAX
9550             AND @#1F
9560             CMP @#12
9570             BNE notind
9580             DEC byt1
9590             JSR nxtsearch
9600             INC byt1
9610             LDY @#0D
9620             STY adm
9630             RTS
9640
9650 :notind     AND @#07
9670             CMP @#07
9680             BNE notspecial
9690             TXA
9700             AND @#BF
9710             TAX
9720 :notspecial TXA
9740             LDX @N-1+#80
9750 :nxtcmp     CMP opcode-#80,X
9770             BEQ new
9780             DEX
9790             BMI nxtcmp
9800
9810 :oldsearch  CMP @#D4
9830             BEQ illegal
9840 :nxtsearch  INY
9860             STY adm
9870             LDX @#40
9880 :nxt'try    SEC
9900             LDA byt1
9910             SBC opctab1,Y
9920             SEC
9930             SBC opctab2,X
9940             BNE no'match
9950             STA tmp1
9960             LDY admgrp,X
9970 :nxtror     ROR A
9990             ROR tmp1
10000            DEY
10010            BNE nxtror
10020            LDY adm
10030            AND admtab1,Y
10040            BNE found
10050            LDA admtab2,Y
10060            AND tmp1
10070            BNE found
10080 :no'match  DEX
10100            BNE nxt'try
10110            CPY @#0E
10120            BNE nxtsearch
10130 :illegal   LDX @#11
10150            LDY @#0F
10160            STY adm
10170 :found     LDA lentab,Y
10190            CMP @#0F
10200            BNE storlen
10210            LDA @1
10220 :storlen   AND @#03
10240            STA len
```

```
10250             RTS
10260
10270 :new        LDA len'adm-#80,X
10290             PHA
10300             AND @#8F
10310             STA adm
10320             PLA
10330             LSR A
10340             LSR A
10350             LSR A
10360             LSR A
10370             BRA storlen
10380
10390 :spctab     .BYT 7,4,1
10410
10420 :admchar    .ASC "AY,)X,(@"
10440
10450 :admformat .BYT #FF,#FE,#FF,#F9
10470             .BYT #CF,#FF,#7F,#7F
10480             .BYT #F9,#CF,#B1,#87
10490             .BYT #FF,#B7,#F9,#FF
10500
10510 :opcode     .BYT #34,#3C,#89,#7C
10530             .BYT #1A,#3A,#80,#04
10540             .BYT #0C,#14,#1C,#5A
10550             .BYT #7A,#DA,#FA,#64
10560             .BYT #74,#9C,#9E,#07
10570             .BYT #87,#0F,#8F
10580
10590 :len'adm    .BYT #14,#29,#16,#2B
10610             .BYT #01,#01,#12,#15
10620             .BYT #2C,#15,#2C,#00
10630             .BYT #00,#00,#00,#15
10640             .BYT #14,#2C,#29,#95
10650             .BYT #95,#A0,#A0
10660
10670 :mneml
10680 :mnemr      = mneml+N
10690
10700 :disend     = mnemr+N
10710
10720             .END
10730
10740 REM MNEMONICS
10750 $B+00="BITBITBITJMP"
10760 $B+12="INCDECBRATSB"
10770 $B+24="TSBTRBTRBPHY"
10780 $B+36="PLYPHXPLXSTZ"
10790 $B+48="STZSTZSTZRMB"
10800 $B+60="SMBBBRBBS"
10810
10820 REM COMPRESS
10830 IF ?#28FF&3:1 GOTO 10910
10840 L=P;R=P+N;IF O<>0 L=O;R=O+N
10850 FOR I=0 TO N-1
10860 G=(?B&#1F+1)*B
10870 L?I=G!((B?1&#1F+1)/4)
10880 R?I=(B?1+1)*64+((B?2+1)*2)&#3E
10890 B=B+3;NEXT I
10900
10910 ASM-CONTINUE
10920
10930             .CODE disend
10940             .RAM M+P-C
10950
10960             \ ---------
10970             \ SIMULATOR
10980             \ ---------
10990
11000 :sim        JSR init
11020             BIT mode
11030             BMI upd'pc
11040             JSR simulate
11050 :upd'pc     LDA len
11070             INC A
11080 :upd'pc'    CLC
11100             BPL P+4
11110             DEC pc+1
11120             ADC pc
11130             STA pc
11140             BCC P+4
11150             INC pc+1
11160             RTS
11170
11180 :simulate   LDY @2
11200             LDA (pc)
11210             BNE no'brk
11220
11230 :brk        JSR push'pc
11250             LDA preg
11260             ORA @:00010000
11270             JSR push
11280             ORA @:00000100
11290             STA preg
11300             LDA intvec
11310             LDX intvec+1
11320 :stor'pc    JSR to'ptr1
11340             BRA zero'len
11350
11360 :rti        JSR pull
11380             STA preg
11390             DEC len
11400
11410 :rts        LDX sreg
11430             INX
11440             BEQ end
11450             CPX @#FE
11460             BNE pull'pc
11470             JSR tst'stack
11480             BNE pull'pc
11490             JSR reset'sreg
11500 :end        STZ cmd
11520 :zero'len   LDA @#FF
11540             STA len
11550             RTS
11560
11570 :pull'pc    JSR pull
11590             STA pc
11600             JSR pull
11610             STA pc+1
11620             RTS
11630
11640 :no'brk     CMP @#40
11660             BEQ rti
11670             CMP @#60
11680             BEQ rts
```

```
11690            CMP @#4C
11700            BEQ jmp
11710            CMP @#6C
11720            BEQ jmpi
11730            CMP @#7C
11740            BEQ jmpx
11750            CMP @#20
11760            BNE tst'bra
11770
11780 :jsr       BVS execute
11800            BIT portc
11810            BVC execute
11820            LDA rreg
11830            BEQ jsr'
11840            DEC A
11850            CMP (pc),Y
11860            BCC execute
11870 :jsr'.     JSR push'pc
11890
11900 :jmp'      CLC
11920 :jmpi      LDA (pc),Y
11940            TAX
11950            DEY
11960            LDA (pc),Y
11970            JSR stor'pc
11980            BCS jmp'
11990            RTS
12000
12010 :jmpx      JSR jmp'
12030            LDY xreg
12040            INY
12050            CLV
12060
12070 :jmp       BVC jmp'
12090            LDA @#20 \ JSR
12100            STA byt1
12110            JSR execute
12120            STZ len
12130            BRA rts
12140
12150 :tst'bra   INY
12170            CPX @#80
12180            BEQ bra'spec
12190            CPX @#02
12200            CLC
12210            BNE execute
12220
12230 :bra'normal LDX @4
12250            LDY @#EA \ NOP
12260
12270 :execute'  STX byt2
12290
12300 :bra'spec  STY byt3
12320
12330 :execute   PHP
12350            TSX
12360            STX tmp1
12370            LDX sreg
12380            TXS
12390            LDA preg
12400            PHA
12410            LDA areg
12420            LDX xreg
12430            LDY yreg
12440            PLP
12450            JMP byt1
12460
12470 :monbrk    PHP
12490            SEC
12500            .BYT skip2
12510 :back      PHP
12530            CLC
12540            STY yreg
12550            STX xreg
12560            STA areg
12570            PLA
12580            STA preg
12590            TSX
12600            STX sreg
12610            LDX tmp1
12620            TXS
12630            ROR A
12640            PLP
12650            RTS
12660
12670 :branch    LDX tmp1
12690            TXS
12700            PLP
12710            LDY @1
12720            BCC P+3
12730            INY
12740            LDA (pc),Y
12750            JMP upd'pc'
12760
12770 :push'pc   TYA
12790            CLC
12800            ADC pc
12810            PHA
12820            LDA pc+1
12830            ADC @0
12840            JSR push
12850            PLA
12860
12870 :push      LDX sreg
12890            STA stack,X
12900            DEC sreg
12910            RTS
12920
12930 :pull      INC sreg
12950            LDX sreg
12960            LDA stack,X
12970            RTS
12980
12990 :tst'stack LDX stack+#FF
13010            CPX @(jmp1-1)/256
13020            BNE ret7
13030            LDX stack+#FE
13040            CPX @(jmp1-1)%256
13050 :ret7      RTS
13070
13080 :init      LDY len
13100 :init'     INY
13120            LDA dummy,Y
13130            STA byt1,Y
13140            CPY @8
13150            BCC init'
```

```
13160 :dummy        RTS \ BYT1
13180               NOP \ BYT2
13190               NOP \ BYT3
13200               JMP back
13210               JMP branch
13220
13230 :simend
13240
13250              .END
13260
13270  RETURN
13280
13290 tREM DISASS TEST
13300  PRINT $12$14'''''''''''''''
13310  W=#2800;U=0;V=ptr1
13320  FOR I=#00 TO #0F
13330  FOR J=#00 TO #F0 STEP #10
13340  ?W=J+I;W?1=?W+1;W?2=?W+2
13350  !V=W;A=1;LINK xdisass';P.'
13360  FOR Q=#81C0 TO #81DC STEP 4
13370  U=U+!Q;NEXT Q;NEXT J;NEXT I
13380  IF U=#AEFF58F9 PRINT"OK"';R.
13390  PRINT $7"FOUT"';RETURN
```

SOURCE: 2900 68F8

OBJECT: MONBEG 7800
MONEND 7C8A
DISEND 7E79
SIMEND 7FCD

AUTEUR: J.JOBSE
DUINWEG 35
4356 AP OOSTKAPELLE

DATUM: AUGUSTUS 1986.

```
  10  REM ----------
  20  REM NMON32 4.2
  30  REM ----------
  40
  50  ROM1;\ SALFAA C2.0
  60
  70  C=#7800;\ CODE (CA 1.75K)
  80  M=#7800;\ RAM
  90  Z=#A0  ;\ ZP (7)
 100  A=#012F;\ ABS (17)
 110
 120  PASS0;GOSUB a
 130  PASS1;GOSUB a
 140
 150  PRINT "MONBEG"&monbeg,'
 160  PRINT "MONEND"&monend,'
 170  PRINT "DISEND"&disend,'
 180  PRINT "SIMEND"&simend,''
 190
 200  INPUT "DIS TEST Y/N "$TOP
 210  IF C=M IF $TOP="Y" GOSUB t
 220  END
 230
 240 aASM-BEGIN
 250
 260             .OPTION :01000000
 270             .CODE C
 280             .RAM M
 290
 300             \ -----------
 310             \ DECLARATIONS
 320             \ -----------
 330
 340 :ack   = #06
 350 :bell  = #07
 360 :lf    = #0A
 370 :cls   = #0C
 380 :cr    = #0D
 390 :home  = #1E
 400 :eot   = #EA
 410 :dept  = #40
 420 :skip1 = #24
 430 :skip2 = #2C
 440
 450 :curpos = #E0
 460 :cursor = #E1
 470 :pagmod = #E6
 480
 490 :ptr1 = Z+0
 500 :ptr2 = Z+2
 510 :ptr3 = Z+4
 520 :len  = Z+6
 530
 540 :pc   = ptr1
 550 :tmp1 = ptr3
 560 :tmp2 = ptr3+1
 570 :tmp3 = len
 580
 590 :cmd  = A+0
 600 :byt1 = A+1
 610 :byt2 = A+2
 620 :byt3 = A+3
 630 :jmp1 = A+4
 640 :jmp2 = A+7
 650 :preg = A+10
 660 :sreg = A+11
 670 :yreg = A+12
 680 :xreg = A+13
 690 :areg = A+14
 700 :rreg = A+15
 710 :mode = A+16
 720
 730 :tmp4  = jmp1
 740 :page  = jmp1+1
 750 :adm   = jmp1+2
 760 :pcsav = jmp2+1
 770
 780 :cmdbuf = #0100
 790 :stack  = #0100
 800 :stack1 = #0200-dept
 810 :stack2 = #0200-2*dept
 820
 830 :screen    = #8000
 840 :portb     = #B001
 850 :portc     = #B002
 860 :getcmd    = #CD0F
 870 :mnetabl   = #F154
 880 :mnetabr   = #F194
 890 :admtab1   = #F1D5
 900 :admtab2   = #F1E4
 910 :opctab1   = #F1F3
 920 :lentab    = #F202
 930 :opctab2   = #F210
 940 :admgrp    = #F250
 950 :decadr    = #F668
 960 :incadr    = #F671
 970 :pradrsp   = #F7F1
 980 :prbytsp   = #F7FA
 990 :prspace   = #F7FD
1000 :prbyt     = #F802
1010 :prnibl    = #F808
1020 :prbit     = #F813
1030 :getbufch  = #F875
1040 :asctohex  = #F87E
1050 :getadr    = #F893
1060 :incadrcmp= #FA08
1070 :cmpadr    = #FA0E
1080 :invert    = #FD44
1090 :frame     = #FE66
1100 :scankb    = #FE71
1110 :rdchar    = #FFE3
1120 :rdecho    = #FFE6
1130 :prascii   = #FFE9
1140 :prcrlf    = #FFED
1150 :prchar    = #FFF4
1160 :starcmd   = #FFF7
1170 :intvec    = #FFFE
1180
1190             \ -------
1200             \ MONITOR
1210             \ -------
1220
1230 :monbeg
1240
1250             \ fixed entry's:
1260
```

```
1270             JMP start \ cold
1280             JMP nxtcmd \ warm
1290             JMP disass
1300             JMP search
1310             .BYT Z \ zp
1320             .WRD A \ abs
1330
1340 :start      JSR zero
1360             LDA @#F8
1370             STA rreg
1380
1390 :nxtcmd     CLD
1410             LDA #52
1420             PHA
1430             LDX #07
1440             LDA @":"
1450             JSR getcmd
1460             STX #07
1470             PLA
1480             STA #52
1490             JSR clrmodpag \ X=0
1500
1510             LDY @#FF
1520 :nxtbufch   JSR getbufch
1540             CMP @cr
1550             BEQ nxtcmd
1560             CMP @#21
1570             BCC nxtbufch
1580             CMP @"*"
1590             BNE no'star
1600
1610 :nxtmov     INY
1630             LDA cmdbuf,Y
1640             STA cmdbuf,X
1650             INX
1660             CMP @cr
1670             BNE nxtmov
1680             JSR starcmd
1690             JMP nxtcmd
1700
1710 :no'star    AND @#DF
1730             STA cmd
1740             TAX
1750             JSR getbufch
1760             CMP @"."
1770             BEQ cmd'srch
1780             CMP @"="
1790             BEQ cmd'srch+1
1800             DEY
1810 :cmd'srch   TXA
1830             LDX @(adrtab-cmdtab-1)&#FF
1840             CMP cmdtab,X
1850             BEQ cmd'fnd
1860             DEX
1870             BPL cmd'srch+3
1880
1890 :cmd'err    JSR pr'com'err
1910             JMP nxtcmd
1920
1930 :cmd'fnd    TXA
1950             PHA
1960             CMP @9 \ =
1970             BEQ getadr3
1980             JSR getadr2
1990             STA tmp4
2000             CPX @","
2010             BNE P+5
2020             JSR setpag
2030             ASL A
2040             BEQ default
2050             LDA ptr2
2060             LDX ptr2+1
2070             JSR to'ptr1
2080 :default    JSR getadr2
2100             TAX
2110             BEQ getadr3
2120             ASL cmd
2130 :getadr3    LDX @ptr3
2150             JSR getadr'
2160             PLA
2170             CMP @5
2180             BCS skp'inc
2190             CMP @3
2200             BCS P+6
2210             LDX ptr3+2
2220             BEQ cmd'err
2230             LDX @(ptr2+#A6)&#FF
2240             JSR incadr
2250 :skp'inc    ASL A
2270             TAX
2280             LDA pagmod
2290             PHA
2300             ORA @#80 \ off
2310             STA pagmod
2320             JSR exec'cmd
2330             PLA
2340             STA pagmod
2350             JMP nxtcmd
2360
2370 :exec'cmd   LDA adrtab+1,X
2390             PHA
2400             LDA adrtab,X
2410             PHA
2420             LDY @0
2430             RTS \ C=0
2440
2450 :getadr2    LDX @ptr2
2470 :getadr'    JSR getbufch
2490             CMP @"#"
2500             BEQ getadr'
2510             JSR getadr
2520             LDX cmdbuf,Y
2530             CPX @cr
2540             BNE P+3
2550             DEY
2560             RTS
2570
2580 :fill       LDA ptr3
2600             STA tmp3
2610             LDA ptr1
2620             LDX ptr1+1
2630             STA ptr3
2640             STX ptr3+1
2650             BCC next
2660
2670 :copy
```

```
2680 :verify      ASL cmd
2700 :next        LDA (ptr1),Y
2720              BIT cmd
2730              BVS compare
2740              BPL store
2750              LDA tmp3
2760 :store       STA (ptr3),Y
2780 :compare     CMP (ptr3),Y
2800              BEQ cmp'ok
2810              LDX @ptr3
2820              JSR pradrsp
2830              LDA (ptr3),Y
2840              JSR prhexasc
2850              JSR control
2860              BCS ret0
2870 :cmp'ok      JSR incptr31
2890              BCC next
2900 :ret0        RTS
2920
2930 :nxtdis      JSR control'
2950              BCS ret1
2960 :dis         JSR dis'
2980              LDY len
2990              INY
3000              JSR updptr1
3010              BCC nxtdis
3020 :ret1        RTS
3040
3050 :nxthex      JSR control
3070              BCS ret1
3080 :hex         LDY @3
3100              STY len
3110              JSR pradrbyts
3120              LDY @0
3130 :nxtasc      JSR prspace
3150              LDA (ptr1),Y
3160              JSR pr'asc
3170              INY
3180              CPY @4
3190              BNE nxtasc
3200              JSR updptr1
3210              BCC nxthex
3220              BCS crlf
3230
3240 :prhexasc    PHA
3260              JSR prbytsp
3270              PLA
3280 :pr'asc      CMP @" "
3300              BCC P+6
3310              CMP @#7F
3320              BCC P+4
3330              LDA @"."
3340              JMP prchar
3350
3360 :no'ram      LDA @bell
3380              JSR prchar
3390 :nxtmem      JSR prcrlf
3410 :memory      JSR pr'ptr1
3430              LDA (ptr1),Y
3440              JSR prhexasc
3450              JSR prspace
3460 :rdmem       JSR rwbyte
3480              BCS no'hex
3490              ORA tmp1
3500              STA (ptr1),Y
3510              CMP (ptr1),Y
3520              BNE no'ram
3530 :nextmem     JSR incptr1
3550              BEQ nxtmem
3560 :no'hex      AND @#DF
3580              CMP @"N"
3590              BEQ nextmem
3600              CMP @"P"
3610              BNE no'prev
3620              LDX @(ptr1+#A6)&#FF
3630              JSR decadr
3640              BCS nxtmem
3650 :no'prev     CMP @cr
3670              BNE rdmem
3680 :crlf        JMP prcrlf
3700
3710 :rwbyte      JSR rwnibl
3730              BCS tst'sp
3740              ASL A
3750              ASL A
3760              ASL A
3770              ASL A
3780              STA tmp1
3790
3800 :rwnibl      JSR rdnibl
3820              BCS ret2
3830              PHA
3840              JSR prnibl
3850              PLA
3860              CLC
3870 :ret2        RTS
3890
3900 :tst'sp      CMP @" "
3920              SEC
3930              BNE ret2
3940              JSR prchar
3950              JSR rdecho
3960              STA tmp1
3970              CLC
3980              RTS
3990
4000 :rdnibl      JSR rdchar
4020              CMP @"G"
4030              BCS ret2
4040              CMP @"A"
4050              BCS P+6
4060              CMP @CH"9"+1
4070              BCS ret2
4080              JMP asctohex
4090
4100 :step        JSR prepare
4120              JMP tst'cmd
4130
4140 :trace       JSR prepare
4160              BPL clr'scr
4170
4180 :go'on1      LDA portb
4200              AND @:01010000 \ CTRL+Q
4210              BEQ return'
4220              LDX @ptr1
4230              JSR cmpadr
```

```
4240               STA screen+#1F
4250               BNE go'on2
4260               LSR cmd
4270               JSR ack'crlf
4280               LDA cmd
4290               CMP @"T"
4300               BNE tst'cmd
4310 :clr'scr      LDA @cls
4330               JSR prchar
4340               JSR coff
4350               JMP tst'cmd
4360
4370 :go'on2       JSR search
4390 :go'on3       LDA sreg
4410               CMP @#0103-dept
4420               BCC stack'err
4430               LDX adm
4440               CPX @#0F
4450               BEQ opcode'err
4460               JSR sim
4470 :tst'cmd      LDA cmd
4490               BMI go'on1
4500 :return'      BEQ return
4520               CMP @"T"
4530               BEQ nxt'trace
4540
4550 :nxtstep      JSR prcrlf
4570               JSR printreg+3
4580 :nxtstep'     JSR clrmodpag
4600               JSR control'
4610               BCC go'on3
4620               BCS return
4630
4640 :nxt'trace    JSR upd'screen
4660               JMP nxtstep'
4670
4680 :stack'err    JSR ack'crlf
4700              .ASC "STACK "
4710               BCC pr'err
4720
4730 :opcode'err   JSR ack'crlf
4750              .ASC "OPCODE ",eot
4760
4770 :pr'err       JSR print
4790              .ASC bell,"ERROR",eot
4800
4810 :pr'at        JSR print
4830              .ASC " AT ",eot
4840               JSR pr'ptr1
4850
4860 :return       JSR con
4880               JSR ack'crlf
4890               LDA ptr1
4900               LDX ptr1+1
4910               STA pcsav
4920               STX pcsav+1
4930
4940 :swap         CLC
4960               LDX @dept-1
4970               LDY stack1,X
4980               LDA stack2,X
4990               STA stack1,X
5000               TYA
5010               STA stack2,X
5020               DEX
5030               BPL swap+3
5040               TSX
5050               TXA
5060               BCC P+4
5070               SBC @2*dept+1
5080               ADC @dept
5090               TAX
5100               TXS
5110               RTS
5120
5130 :go           JSR prepare
5150               LDY @#20 \ JSR
5160               LDX sreg
5170               INX
5180               BEQ stack'ok
5190               LDX tmp4
5200               BEQ continue
5210               JSR reset'sreg
5220               BNE stack'ok
5230 :continue     JSR tst'stack
5250               BNE stack'err
5260               LDY @#4C \ JMP
5270 :stack'ok     STY byt1
5290               LDY @0
5300               BVS no'brkpnt
5310 :nxtpush      LDA (ptr2),Y
5330               PHA
5340               LDA brkpnt,Y
5350               STA (ptr2),Y
5360               INY
5370               CPY @3
5380               BNE nxtpush
5390               DEY
5400 :no'brkpnt    JSR init'
5420               LDX @realtime%256
5430               LDY @realtime/256
5440               STY byt3
5450               JSR execute'
5460               BVS return
5470               ASL A
5480               LDY @2
5490 :nxtpull      PLA
5510               STA (ptr2),Y
5520               DEY
5530               BPL nxtpull
5540               BCC return
5550               JSR pull'pc
5560               LDA @#FE \ -2
5570               JSR upd'pc'
5580               JSR ack'crlf
5590              .ASC "BREAK",eot
5600               JMP pr'at
5610
5620 :brkpnt       JSR monbrk
5640
5650 :realtime     JMP (pc)
5670
5680 :prepare      SEC
5700               JSR swap+1
5710               LDA tmp4
5720               BNE ret3
```

```
5730 :ldpc        LDA pcsav
5750              LDX pcsav+1
5760 :to'ptr1     STA ptr1
5780              STX ptr1+1
5790 :ret3        BIT cmd
5810              RTS
5820
5830 :printreg    JSR ldpc
5850              JSR prheader'
5860 :pr'val      JSR pr'values
5880 :dis'        JSR disass
5900              JMP prcrlf
5910
5920 :upd'screen  JSR prheader
5940              LDY @#80
5950              JSR frame
5960 :nxtscroll   LDA screen-#20,Y
5980              STA screen-#40,Y
5990              LDA screen+#A0,Y
6000            . STA screen+#80,Y
6010              INY
6020              BNE nxtscroll
6030              JSR prlf5
6040              JSR pr'values
6050              JSR prlf5
6060              BEQ dis'
6070
6080 :prlf5       LDX @5
6100              LDA @lf
6110              JSR prchar
6120              DEX
6130              BNE P-4
6140              RTS
6150
6160 :prheader    LDA @home
6180              JSR prchar
6190 :prheader'   JSR print
6210             .ASC " PC   A  X  Y  S"
6220             .ASC " NV BDIZC  R"
6230 :prcrlf'     BCC prcrlf''
6250
6260 :pr'values   JSR pr'ptr1
6280              LDX @4
6290 :nxtreg      LDA preg,X
6310              JSR prbytsp
6320              DEX
6330              BNE nxtreg
6340              LDA preg
6350              LDY @8
6360 :nxtbin      ASL A
6380              PHA
6390              TXA
6400              JSR prbit
6410              PLA
6420              DEY
6430              BNE nxtbin
6440              JSR prspace
6450              LDA rreg
6460              JSR prbyt
6470 :prcrlf''    JMP prcrlf
6490
6500 :zero        LDA @0
6520              STA areg
6530              STA xreg
6540              STA yreg
6550              LDA @:00110100
6560              STA preg
6570 :reset'sreg  LDA @#FF
6590              STA sreg
6600              RTS
6610
6620 :asign       LDX @(cmdtab-regtab-1)&#FF
6640              LDA cmd
6650 :nxt'asign   CMP regtab,X
6670              BEQ asign'fnd
6680              DEX
6690              BPL nxt'asign
6700
6710 :pr'com'err  JSR print
6730             .ASC bell,"COMMAND ERROR"
6740              BCC prcrlf''
6750
6760 :asign'fnd   LDA ptr3
6780              STA preg,X
6790              RTS
6800
6810 :quit        PLA
6830              PLA
6840              PLA
6850              STA pagmod
6860              RTS
6870
6880 :control     JSR prcrlf
6900 :control'    JSR scankb
6920              CPY @cr
6930              BEQ ret4
6940              LDX page
6950              BNE setpag+2
6960              CPY @#2E \ N
6970              BEQ ret4-1
6980              TYA \ SP
6990              BEQ space'bar
7000              LDA @:01000000
7010              CPY @#32 \ R
7020              BEQ upd'mode
7030              CPY @#33 \ S
7040              BNE control'
7050              ASL A
7060 :upd'mode    STA mode
7080              CLC
7090              RTS
7100 :space'bar   JSR prcrlf
7120 :setpag      LDX @15
7140              DEX
7150              STX page
7160              CLC
7170 :ret4        RTS
7190
7200 :clrmodpag   LDX @0
7220              STX mode
7230              BEQ setpag+3
7240
7250 :pr'ptr1     LDX @ptr1
7270              JMP pradrsp
7280
7290 :incptr31    LDX @(ptr3+#A6)&#FF
```

```
7310           JSR incadr
7320 :incptr1  LDY @1
7340 :updptr1  LDX @ptr1
7360           JSR incadrcmp
7370           BEQ ret5
7380           DEY
7390           BNE updptr1+2
7400           CLC
7410 :ret5     RTS
7430
7440 :ack'crlf JSR print
7460          .ASC ack,cr,eot
7470 :print    PLA
7490           STA ptr3
7500           PLA
7510           STA ptr3+1
7520           LDY @0
7530 :nxtpr    INC ptr3
7550           BNE P+4
7560           INC ptr3+1
7570           LDA (ptr3),Y
7580           BMI endpr
7590           JSR prascii
7600           JMP nxtpr
7610 :endpr    CLC
7630           JMP (ptr3)
7640
7650 :con      LDA @#B0
7670          .BYT skip2
7680 :coff     LDA @0
7700           CMP cursor
7710           BEQ ret5
7720           STA cursor
7730           LDY curpos
7740           JMP invert
7750
7760 :regtab  .ASC "PSYXAR"
7780
7790 :cmdtab  .ASC "CVFHDSTGM=PZQ"
7810
7820 :adrtab  .WRD copy-1
7840          .WRD verify-1
7850          .WRD fill-1
7860          .WRD hex-1
7870          .WRD dis-1
7880          .WRD step-1
7890          .WRD trace-1
7900          .WRD go-1
7910          .WRD memory-1
7920          .WRD asign-1
7930          .WRD printreg-1
7940          .WRD zero-1
7950          .WRD quit-1
7960
7970 :monend
7980
7990          \ -----------
8000          \ DISASSEMBLER
8010          \ -----------
8020
8030 :disass   JSR search
8050           TXA
8060           PHA
8070           JSR pradrbyts
8080           LDA spctab,Y
8090           TAY
8100           JSR pryspc
8110
8120           PLA
8130           TAX
8140           LDA mnetabr,X
8150           LDY mnetabl,X
8160           STA tmp1
8170           STY tmp2
8180           LDY @3
8190 :nxtchar  LDX @5
8210           LDA @0
8220 :nxtrol   ROL tmp1
8240           ROL tmp2
8250           ROL A
8260           DEX
8270           BNE nxtrol
8280           ADC @#3F
8290           JSR prchar
8300           DEY
8310           BNE nxtchar
8320           JSR prspace
8330
8340           LDX adm
8350           LDY @11
8360           STY tmp1
8370           LDY @7
8380           LDA admformat,X
8390           STA tmp2
8400 :nxtbit   CPY @5
8420           BNE nxtasl
8430           TXA
8440           BNE skiprel
8450
8460           LDA byt2
8470           BPL P+3
8480           DEX
8490           ADC ptr1
8500           PHA
8510           TXA
8520           ADC ptr1+1
8530           TAX
8540           PLA
8550           CLC
8560           ADC len
8570           STA byt2
8580           TXA
8590           ADC @0
8600           STA byt3
8610           LDX @2
8620           BNE prhex
8630
8640 :skiprel  LDX len
8660           BEQ nxtasl
8670 :prhex    LDA @"#"
8690           JSR prchar
8700           DEC tmp1
8710 :nxtbyt'  LDA byt1,X
8730           JSR prbyt
8740           DEC tmp1
8750           DEC tmp1
```

```
8760            DEX
8770            BNE nxtbyt'
8780 :nxtasl    ASL tmp2
8800            BCS skipchar
8810            LDA admchar,Y
8820            JSR prchar
8830            DEC tmp1
8840 :skipchar  DEX
8860            DEY
8870            BPL nxtbit
8880            LDY tmp1
8890
8900 :pryspc    INY
8920            DEY
8930            BEQ ret6
8940            JSR prspace
8950            JMP pryspc+1
8960
8970 :pradrbyts JSR pr'ptr1
8990            LDY @#FF
9000 :nxtbyt    INY
9020            LDA (ptr1),Y
9030            JSR prbytsp
9040            CPY len
9050            BNE nxtbyt
9060 :ret6      RTS
9080
9090 :search    LDY @2
9110            LDA (ptr1),Y
9120            STA byt1,Y
9130            DEY
9140            BPL search+2
9150            CMP @#D4
9160            BEQ illegal
9170 :nxtsearch INY
9190            STY adm
9200            LDX @#40
9210 :nxt'try   SEC
9230            LDA byt1
9240            SBC opctab1,Y
9250            SEC
9260            SBC opctab2,X
9270            BNE no'match
9280            STA tmp1
9290            LDY admgrp,X
9300 :nxtror    ROR A
9320            ROR tmp1
9330            DEY
9340            BNE nxtror
9350            LDY adm
9360            AND admtab1,Y
9370            BNE found
9380            LDA admtab2,Y
9390            AND tmp1
9400            BNE found
9410 :no'match  DEX
9430            BNE nxt'try
9440            CPY @#0E
9450            BNE nxtsearch
9460 :illegal   LDX @#11
9480            LDY @#0F
9490            STY adm
9500 :found     LDA lentab,Y
9520            CMP @#0F
9530            BNE storlen
9540            LDA @1
9550 :storlen   AND @#03
9570            STA len
9580            RTS
9590
9600 :spctab    .BYT 7,4,1
9620
9630 :admchar   .ASC "AY,)X,(@"
9650
9660 :admformat .BYT #FF,#FE,#FF,#F9
9680            .BYT #CF,#FF,#7F,#7F
9690            .BYT #F9,#CF,#B1,#B7
9700            .BYT #FF,#B7,#F9,#FF
9710
9720 :disend
9730
9740            \ ---------
9750            \ SIMULATOR
9760            \ ---------
9770
9780 :sim       JSR init
9800            BIT mode
9810            BMI upd'pc
9820            JSR simulate
9830 :upd'pc    LDA len
9850            CLC
9860            ADC @1
9870 :upd'pc'   CLC
9890            BPL P+4
9900            DEC pc+1
9910            ADC pc
9920            STA pc
9930            BCC P+4
9940            INC pc+1
9950            RTS
9960
9970 :simulate  LDY @2
9990            LDA byt1
10000           BNE no'brk
10010
10020 :brk      JSR push'pc
10040           LDA preg
10050           ORA @:00010000
10060           JSR push
10070           ORA @:00000100
10080           STA preg
10090           LDA intvec
10100           LDX intvec+1
10110 :stor'pc  JSR to'ptr1
10130 :zero'len LDA @#FF
10150           STA len
10160           RTS
10170
10180 :rti      JSR pull
10200           STA preg
10210 :rts'     DEC len
10230
10240 :rts      LDX sreg
10260           INX
10270           BEQ end
10280           CPX @#FE
```

```
10290            BNE pull'pc
10300            JSR tst'stack
10310            BNE pull'pc
10320            JSR reset'sreg
10330 :end       LDA @0
10350            STA cmd
10360            BEQ zero'len
10370
10380 :pull'pc   JSR pull
10400            STA pc
10410            JSR pull
10420            STA pc+1
10430            RTS
10440
10450 :no'brk    CMP @#40
10470            BEQ rti
10480            CMP @#60
10490            BEQ rts
10500            CMP @#4C
10510            BEQ jmp
10520            CMP @#6C
10530            BEQ jmpi
10540            CMP @#20
10550            BNE tst'bra
10560
10570 :jsr       BVS execute
10590            BIT portc
10600            BVC execute
10610            LDA rreg
10620            BEQ jsr'
10630            LDA (pc),Y
10640            CMP rreg
10650            BCS execute
10660 :jsr'      JSR push'pc
10680
10690 :jmp'      CLC
10710 :jmpi      LDA (pc),Y
10730            TAX
10740            DEY
10750            LDA (pc),Y
10760            JSR stor'pc
10770            BCS jmp'
10780            RTS
10790
10800 :jmp       BVC jmp'
10820            LDA @#20 \ JSR
10830            STA byt1
10840            JSR execute
10850            DEC len
10860            BNE rts'
10870
10880 :tst'bra   CPX @#02
10900            BNE execute
10910 :bra       LDX @4
10930
10940 :execute'  STX byt2
10960
10970 :execute   PHP
10990            TSX
11000            STX tmp1
11010            LDX sreg
11020            TXS
11030            LDA preg
11040            PHA
11050            LDA areg
11060            LDX xreg
11070            LDY yreg
11080            PLP
11090            JMP byt1
11100
11110 :monbrk    PHP
11130            SEC
11140           .BYT skip2
11150 :back      PHP
11170            CLC
11180            STY yreg
11190            STX xreg
11200            STA areg
11210            PLA
11220            STA preg
11230            TSX
11240            STX sreg
11250            LDX tmp1
11260            TXS
11270            ROR A
11280            PLP
11290            RTS
11300
11310 :branch    LDX tmp1
11330            TXS
11340            PLP
11350            LDY @1
11360            LDA (pc),Y
11370            JMP upd'pc'
11380
11390 :push'pc   TYA
11410            CLC
11420            ADC pc
11430            PHA
11440            LDA pc+1
11450            ADC @0
11460            JSR push
11470            PLA
11480
11490 :push      LDX sreg
11510            STA stack,X
11520            DEC sreg
11530            RTS
11540
11550 :pull      INC sreg
11570            LDX sreg
11580            LDA stack,X
11590            RTS
11600
11610 :tst'stack LDX stack+#FF
11630            CPX @(jmp1-1)/256
11640            BNE ret7
11650            LDX stack+#FE
11660            CPX @(jmp1-1)%256
11670 :ret7      RTS
11690
11700 :init      LDY len
11720 :init'     INY
11740            LDA dummy,Y
11750            STA byt1,Y
11760            CPY @8
```

```
11770              BCC init'
11780 :dummy       RTS \ BYT1
11800              NOP \ BYT2
11810              NOP \ BYT3
11820              JMP back
11830              JMP branch
11840
11850 :simend
11860
11870             .END
11880
11890  RETURN
11900
11910 tREM DISASS TEST
11920  PRINT $12$14'''''''''''''''
11930  W=#2800;U=0;V=ptr1
11940  FOR I=#00 TO #0F
11950  FOR J=#00 TO #F0 STEP #10
11960  ?W=J+I;W?1=?W+1;W?2=?W+2
11970  !V=W;LINK disass;P.'
11980  FOR Q=#81C0 TO #81DC STEP 4
11990  U=U+!Q;NEXT Q;NEXT J;NEXT I
12000  IF U=#E8B64808 PRINT"OK"';R.
12010  PRINT $7"FOUT"';RETURN
```

SOURCE: 2900 61EA

OBJECT: MONBEG 7B00
MONEND 7C8A
DISEND 7DA7
SIMEND 7EE8

AUTEUR: J.JOBSE
DUINWEG 35
4356 AP OOSTKAPELLE

DATUM: AUGUSTUS 1986.

OVERZICHT MONITOR WERKRUIMTE

```
#00A0..#00A6  zero page ram (7)
#0100..#012E  commando buffer (47)
#012F..#013F  absolute ram (17)
#0180..#01BF  stack2 (64)
#01C0..#01FF  stack1 (64)
```

INDELING ZP EN ABS WERKRUIMTE

area	rel	prim	sec	area
P	Z+0	ptr1	pc	
P	Z+1	ptr1+1	pc+1	
P	Z+2	ptr2		
P	Z+3	ptr2+1		
P	Z+4	ptr3	tmp1	D
P	Z+5	ptr3+1	tmp2	D
	Z+6	len	tmp3	D
	A+0	cmd		D
E	A+1	byt1		D
E	A+2	byt2		D
E	A+3	byt3		D
E	A+4	jmp1	tmp4	D
E	A+5	jmp1+1	page	D
E	A+6	jmp1+2	adm	D
E	A+7	jmp2		
E	A+8	jmp2+1	pcsav	R
E	A+9	jmp2+2	pcsav+1	R
	A+10	preg		R
	A+11	sreg		R
	A+12	yreg		R
	A+13	xreg		R
	A+14	areg		R
	A+15	rreg		R
	A+16	mode		R

rel = relatief adres
prim = primair gebruik
sec = secundair gebruik

P = Pointer area
E = Execution area
D = Data area
R = Register area

CHECKSUM (C=#7800;Z=#A0;A=#12F)

CMON32 4.2 SUM = #7BD7
NMON32 4.2 SUM = #21EB

BIT GEORIENTEERDE SIGNALERINGEN

mode: bit0 = mode flag
bit6 = realtime flag
bit7 = skip flag

N 7	V 6		C 0	
0	0	next one	0	nmos mode
0	1	realtime exec	1	cmos mode
1	0	skip one		
1	1	n.v.t.		

cmd: bit6 = break/display flag (inverse)
bit7 = break/display flag

N 7	V 6	
0	0	n.v.t.
0	1	geen y-adres opgegeven
1	0	wel y-adres opgegeven
1	1	n.v.t.

commando	ascii	hex	binair
STEP	S	#53	01010011
TRACE	T	#54	01010100
GO	G	#47	01000111

cmd: bit6 = verify flag
bit7 = fill flag

N 7	V 6	
0	0	copy mode
0	1	verify mode
1	0	fill mode
1	1	n.v.t.

commando	ascii	hex	binair
COPY	C	#43	01000011
VERIFY	V	#56	01010110
FILL	F	#46	01000110

(n.v.t. = niet van toepassing)

AFKORTINGEN

```
ack  = acknowledge
lf   = line feed
cls  = clear screen
cr   = carriage return
eot  = end of text (NOP)
ptr  = pointer
len  = length of instruction
pc   = program counter
tmp  = temporary storage
cmd  = command
reg  = register
mne  = mnemonic
adm  = adressing mode
opc  = opcode
pr   = print
sp   = space
rd   = read
nxt  = next
srch = search
fnd  = found
skp  = skip
ret  = return
tst  = test
ld   = load
upd  = update
tab  = table
```

SKIP INSTRUCTIES

```
skip1 = #24 = BIT zp
skip2 = #2C = BIT abs
```

Deze BIT-instructies worden "misbruikt" als single byte instructies welke resp. 1 of 2 bytes van de volgende instructie overslaan, waarbij alleen de N/V/Z flags aangetast worden.

BEELDSCHERM AFHANKELIJKE ZAKEN

Declaraties:

curpos,cursor,pagmod
screen,invert,frame

Routines:

```
upd'screen = update screen (trace)
con        = cursor on
coff       = cursor off
setpag     = set page length (15)
```

Iinstructies:

```
STA screen+#1F  simulator indicatie
```

KEYBOARD AFHANKELIJKE ZAKEN

Declaraties:

portb,portc,scankb

Routines:

```
control   wait for CR/SP/N/R/S
          CR --> C=1
```

Instructies:

```
LDA portb          go'on1
AND @:01010000     CTRL+Q

BIT portc          jsr
BVC execute        REPT
```

CLEVER CALIGRAPHY

Dit calligraphy programma print, m.b.v. een Dot-Matrix printer, calligraphy letters in een italiaansachtige stijl. Het orginele programma is gemaakt door Mike Turner voor een BBC-Computer. Dit staat in" ACORN USER ", september 1986. Aangezien onze ATOM (bijna) alles kan, draait dit programma, met enige aanpassing ook. Helaas staat het programma barstensvol data's (Commodore 64 stijl), hiervan houden de ATOM's niet (vrij langzaam). Dus ik heb het geprobeerd te schrijven in machinetaal. Gezien dit programma is dit aardig gelukt, hiermee stel ik niet dat het programma perfect is, maar het is misschien een aanzet tot perfectionering door mede atomisten. Het programma is bijna 4K lang, en gebruikt geen boxen dus het is op elke ATOM te draaien. Voor suggestie's kan je altijd bellen.
Het programma is opgebouwd uit zeven tabellen en een printroutine. De eerste tabel is voor de letterpositie, de tweede is voor de breedte van de letter en de derde tabel is voor de plaats in de volgende tabellen. Deze worden gebruikt voor de letteropbouw, want een letter is opgebouwd uit 4 X 8 bits.
Na een run voer je een tekst in (ong.14 karakters) en het verschijnt op de printer. Zelf heb ik een klein thermisch printertje, de EPSON P-40, met een papierbreedte van 11, 2cm en 256 bits in ch" K " mode, dus waarschijnlijk moeten de printercode's aangepast worden.

Peter Wokke
BOTERSTRAAT 9
1811 HP ALKMAAR
Tel.072-112431

```
 10 REM SOURCE CALLIGRAPHY
 20 REM #2900 - #2D4F
 30 DIM LL(14)
 40 F.I=0 TO 14;LLI=-1;N.
 50 F.I=0 TO 1;P=#2900;P.$21
 60 !P=#4C0A000D;P!4=#32232E49
 70 P!8=#38343139;P!12=#FF0D2E45
 80 P!16=#FF00FFFF;P=P+18;[              \"10LI.#2914;E."
 90:LL0 BRK;NOP
100:LL1 JSR #CD09                        \ INPUT STING ON #140
110     LDX@#00;STX #80;STX #81
120     LDA @21;JSR LL11                 \ PAGE MODE OFF
130:LL2 INX;LDA #0140,X;CMP@#0D          \ RETURN ?
140     BNE LL2;CPX@#0E                  \ 14 CHARACTERS ?
150     BCS LL0                          \ ERROR 20
160     STX #81;DEX
```

```
170:LL3 SEC;LDA #0140,X;SBC@#20
180     TAY;LDA #2A8D,Y                 \ POSISION
190     STA #0140,X;CMP@#50             \ UNKNOWN CHARACTER
200     BCS LL0                         \ ERROR 20
210     TAY;LDA #80;SEC;SBC #2A42,Y     \ WIDTH
220     STA #80;DEX;BPL LL3;CLC;ROR#80  \ CENTER STRING
230     LDA@#02;JSR #FEFB               \ PRINTER ON
240     LDA@#1B;JSR LL11                \ ESC
250     LDA@#41;JSR LL11                \ CH"A"
260     LDA@#08;JSR LL11;JSR LL13       \ 1/9 INCH LINE SPACE
270:LL4 LDY #84;LDA #0140,Y;TAY         \ FIRST TABLE
280     LDA #2AE8,Y;STA#82              \ LOW BIT
290     LDA #2C14,Y;STA#83              \ HIGH BIT
300     JSR LL8;CMP #81;BNE LL4
310     JSR LL12
320:LL5 LDY #84;LDA #0140,Y;TAY         \ SECOND TABLE
330     LDA #2B33,Y;STA#82              \ LOW BIT
340     LDA #2C5F,Y;STA#83              \ HIGH BIT
350     JSR LL8;CMP #81;BNE LL5
360     JSR LL12
370:LL6 LDY #84;LDA #0140,Y;TAY         \ THIRTH TABLE
380     LDA #2B7E,Y;STA#82              \ LOW BIT
390     LDA #2CAA,Y;STA#83              \ HIGH BIT
400     JSR LL8;CMP #81;BNE LL6
410     JSR LL12
420:LL7 LDY #84;LDA #0140,Y;TAY         \ FOURTH TABLE
430     LDA #2BC9,Y;STA#82              \ LOW BIT
440     LDA #2CF5,Y;STA#83              \ HIGH BIT
450     JSR LL8;CMP #81;BNE LL7
460     LDA@#0D;JSR LL11
470     LDA@#03;JSR #FEFB               \ PRINTER OFF
480     LDA@#06;JSR #FEFB;RTS
490:LL8 LDA #2A42,Y;STA #85;LDY@#FF     \ WIDTH
500:LL9 INY;LDA (#82),Y;CMP@#05         \ END OF TABLE
510     BEQ LL10;JSR LL11
520     CPY #85;BCC LL9
530:LL10INY;LDA@#00;JSR LL11            \ FILL WIDTH WITH SPACE'S
540     CPY #85;BNE LL10;INC#84
550     LDA #84;RTS
560:LL11PHA;ROL A;LDA@#03;ROL A         \ SEND PRINTERBIT
570     STA#B003;PLA;ORA@#80;JSR#FEFB
580     LDA@#06;STA #B003;RTS
590:LL12LDA@#0D;JSR LL11                \ CARRIAGE RETURN
600:LL13LDY@#00;STY #84;SEC;LDA@#FF     \ SET SPACE
610     SBC #80;TAX
620     LDA@#1B;JSR LL11                \ ESC
630     LDA@#4B;JSR LL11                \ CH"K"
640     TXA;JSR LL11                    \ TOTAL WIDTH
650     LDA@#00;JSR LL11;LDX #80
660:LL14LDA@#00;JSR LL11;DEX            \ PRINT SPACE
670     BNE LL14;RTS
680];N.;P.$6;E.
```

ENKELE TOEPASSINGEN:

DE A.C.C.
WENST U
PRETTIGE
FEESTDAGEN
&
EEN GELUKKIG
NIEUWJAAR
ATOM NIEUWS

computer special

~~~~~~~~~~~~~~~

ABCDEFGHIJ
KLMNOPQRST
UVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789
! "&'( ).-.~

''''''''''''''''''''''''
~~~~~~~~~~~~~~~

Dit programma is geheel geschreven in P-Charm basic en is z.g. DISK-georienteerd. Het geeft de gebruiker de mogelijkheid om zijn voorraad te beheren. Voor Atomisten zonder disk-operating systeem is dit programma NIET te gebruiken, omdat alle handelingen direct op de diskette worden uitgevoerd. Alleen de sort gebruikt intern geheugen.

GEBRUIK

1 - Opbouwen masterbestand.

Deze optie maakt het mogelijk om of, een nieuw bestand op te bouwen danwel nieuwe records toe te voegen aan een bestaand bestand. Het programma "kijkt" of er een file met de naam MASTER op de diskette staat. Zo niet, dan wordt dit gemeld en er wordt gevraagd of er een nieuw bestand moet worden gealloceerd. Zoja, dan wordt er een file van 1280 bytes (25 records) gealloceerd met de naam "MASTER". Hierna vraagt het programma achtereenvolgens om:

De naam van het artikel.
Het aantal verkocht van dit artikel.
De inkoopprijs.
De verkoopprijs.
Het aantal artikelen in voorraad.
Het aantal artikelen in bestelling bij de Clubwinkel.
De datum.

De omvang van de file (standaard 1280 bytes) kan uiteraard vergroot worden. De enige beperking is het beschikbare interne geheugen omdat er intern wordt gesorteerd. Dit gebeurt vanaf #8200, dus een bestand van 6k is geen probleem.(120 records).

2 - Printen masterbestand.

Hiermee wordt het bestand geprint op het scherm of op de printer.
Records met omschrijving "^" worden geprint met omschrijving "VERVALLEN". (zie Optie 3 en 4)

3 - Muteren masterbestand.

Met deze optie kunt U alle rubrieken in een record veranderen.
Ook wordt hier aangegeven of een record bij het sorteren moet worden verwijderd. Dit gebeurd tijdens het veranderen van de omschrijving van het artikel, door deze te veranderen in een "^" (geinverteerde UP-arrow).

Tijdens het muteren van de rubrieken worden in bepaalde gevallen ook andere velden gemuteerd deze zijn:

a - Als het aantal verkocht wordt gemuteerd wordt ook het aantal in voorraad bijgewerkt.

b - Als de voorraad wordt gemuteerd en het aantal opgegeven is positief dan wordt het aantal bestelde artikelen verminderd met het aantal ontvangen artikelen.

4 - Sorteren masterbestand.

Er wordt eerst een Array gedimensioneerd waarna het bestand wordt ingelezen. Nu wordt het gesorteerd en daarna teruggeschreven naar de diskette. De records met als omschrijving "^" worden niet teruggeschreven en zijn dus verwijderd uit het bestand.De sort is niet zo snel (bubble sort), maar omdat een voorraad-bestand redelijk statisch is, wat artikelnamen betreft, komt het niet zo vaak voor dat men moet sorteren. Dit is alleen nodig nadat er artikelen zijn vervallen of nieuwe artikelen zijn toegevoegd.

5 - Restoren masterbestand.

Deze optie schrijft het backupbestand (master1) terug naar het masterbestand.

6 - Backup masterbestand.

Deze optie maakt een backup van het masterbestand. De backupfile heet MASTER1.

Hopelijk is deze beschrijving voldoende om met het programma te kunnen werken. Dus veel succes ermee !!!

Philip van mourik.

'Me Tarzan, you database'

Dit programma tekent met behulp van de ACORN plaatjes op het scherm. Deze plaatjes kunnen b.v. schema's zijn. Hoe maken we dat?
Eerst moet er een schakelsoft aanwezig zijn. Laadt deze dus eerst in. Nu wordt JOYTEK geladen. Nadat deze wordt gerund, zal worden gevraagd "nieuw of bestaand n/b". Toets nu n in, omdat we de eerste keer geen bestaande tekening hebben. Na <return> zal op het scherm een "tekenvel" met in de kaders allerlei tekens en in het midden een grijze transistor verschijnen. Gebruiken we nu de joystick dan verschuift dit symbool. Met de vuurknop "plotten" we het symbool en wordt deze wit. Men hoort een piepje en het teken staat. Het teken verdwijnt weer, als de grijze en de witte presies opelkaar staan en de vuurknop wordt gedrukt. Een schema met alleen transistoren is ook niet alles. Toets nu een letter in die boven een ander symbooltje staat. B.v. de d. Nu wordt de transistor vervangen door een diode. Deze kan men netzo plotten als de transistor. Andere letter, ander symbool. Nu hebben we ook speciale letters:

@ = draai het symbool om
s = stop het programma
x = voer tekst in; met de cursortoetsen bepaalt men de juiste plaats van de text. dan typt men de text in en sluit dit af met <return>.

Dan hebben we nog de letter y voor het lyntrekken. Drukt men deze in, dan zien we een pijltje. Verplaats dit pijltje naar de plaats vanwaar een lijn getrokken moet worden. Druk op de vuurknop. Verplaats nu het pijltje naar de plaats waar de lijn moet eindigen. Druk op de vuurknop en de lijn staat.
Zo ook de letter z, voor een cirkel. Zet eerst het pijltje op het middelpunt en druk de vuurknop. Dan verplaats je het pijltje naar de buitenkant. Druk op de vuurknop en de cirkel staat.
Als laatste hebben we nog de letter v voor een vierkant of rechthoek. Dit werkt netzo als de cirkel. Eerst de linker onderhoek aangeven en dan de rechter bovenhoek. Kiest men hier de verkeerde volgorde, dan wordt de rechthoek gespiegeld.

Heeft men met lijnen, cirkels en rechtoeken gesloten vlakken gemaakt, dan kan men deze inkleuren. Plaats het symbool (b.v. het pijltje) ergens in zo'n vlak. Toets het cijfer in, dat bij het gewenste patroon staat en het vlak wordt er mee gevuld.
Na het indrukken van de letter s komt de vraag op het scherm "saven j/n". Drukt men n, dan eindigt het programma. Drukt men j, dan wordt de tekening op de cassette of floppy gezet en daarna wordt het programma beeindigd. Deze gesavede tekening kan later weer (bij de vraag "niew of bestaand") weer worden geladen.

Voor mensen zonder joystick: ZIE ONDERAAN DE LISTING!!!!

Veel succes.

```
 1PROGRAM TEK
 2REM NODIG:SCHAKELKAART MET P-CHARM,JOSBOX EN GAGSROM.
 3REM SCHAKELSOFT EN JOYSTICK VOLGENS CLUBAANSLUITING.
10DIMT(32),U(32);P.$12
15$U="***TEKENPROGRAMMA***BY G.T.H.***";P.$U
16IN."NIEUW OF BESTAAND N/B"$T;IF$T="N"G.18
17COS1;*LOAD"TEKBEST"8200
18GRMOD;X=110;Y=100;Z=0;?#E1=0;P.$U
19P."?"'":"'"."'","'"9"'"8"'"7"'"6"'"5"'"4"'"2"'"1"'"0"'
20P."Y V Z @XS W I E D L O A R C N P - O O "$30$10
21MOVE10,22;DRAW10,179;MOVE20,22;DRAW20,179
22FORX=35TO168STEP12;MOVE0,X;DRAW20,X;NEXTX
23MOVE0,179;DRAW300,179;MOVE0,11;DRAW300,11
24MOVE0,22;DRAW300,22;NOSNOW;BASE#40
25CR./P:0#AA#55#AA#55#AA#55#AA#55
26CR./P:1 #CC,0#CC,0#CC,0#CC,0
27CR./P:2 #EE#BB#EE#BB#EE#BB#EE#BB
28CR./P:4 #55#55#55#55#55#55#55#55
29CR./P:5 #55#44#55#11#55#44#55#11
30CR./P:6 #44#88#11#22#44#88#11#22
31CR./P:7 #FF,0#FF,0#FF,0#FF,0
32CR./P:8 #22#11#88#44#22#11#88#44
33CR./P:9 #66#99#99#66#66#99#99#66
34CR./P:10#88,0#22,0#88,0#22,0
35CR./P:11 1,1,1,1,1,1,1#FF
36CR./P:12 #FE#FE#FE,0#EF#EF#EF,0
49CR.TOR,#11,#13,#14,#F8,#14,#12,#11,0,0,0,0,0,0,0,0,0;AS.TOR,9
50CR.WEE,0,0,#7E,#FF,#7E,0,0,0,0,0,0,0,0,0,0
51CR.WES,#10,#38,#38,#38,#38,#38,#38,#10,0,0,0,0,0,0,0
52CR.QTR,#11,#12,#1C,#F8,#14,#12,#11,0,0,0,0,0,0,0,0
55CR.PYL,0,0,0,0,0#80#80#E0,0,0,0,0,0,0,0
60CR.CON,#24,#24,#24,#E7,#24,#24,#24,0,0,0,0,0,0,0,0
61CR.AND,#38,#E4,#22,#23,#22,#E4,#38,0,0,0,0,#80,0,0,0,0
62CR.OR,#70,#CC,#42,#21,#42,#CC,#70,0,0,0,0,0,0,0,0
63CR.INV,0,0,#40,#40,#40,0,0,0,0,0,0,0,0,0,0
64CR.LYN,0,0,0,#F7#15,8,0,0,0,0,0#77#54#88,0,0
65CR.DIO,#22,#32,#3A,#FF,#3A,#32,#22,0,0,0,0,0,0,0,0
66CR.ELC,#2E,#2A,#2A,#EB,#2A,#2A,#2E,0,0,0,0,0,0,0,0
67IM.:TOR,224,9;IM.:CON,208,9;IM.:WEE,192,9
68IM.:AND,176,9;IM.:OR,160,9;IM.:LYN,140,9;IM.:QTR,240,9
69IM.:DIO,128,9;IM.:ELC,112,9;IM.:INV,99,9;IM.:WES,81,9
71FORX=0TO12;PA.12,((X*11)+26),X;NEXTX;PA.12,170;X=110
73bSET9,X,Y;UNSET9
75ATK.(1,2,4,5,6,7,8,9,0)(301,302,304,305,306,307,308,309,310
76ATK.(<,>,:,?)(311,312,313,314)
80ATK.(N,R,C,W,A,O,I,L)(160,170,180,190,200,210,220,230)
85ATK.(D,E,P,Y,Z,V,@,S,X)(240,250,260,270,280,290,350,400,700)
88J. X,Y,G;IFG=0 G.b
100IF Z=1;G.800
101IF Z=2;G.810
102IF Z=3;G.820
103IF Z=4;G.830
```

```
104IF Z=5;G.840
105IF Z=6;G.850
150SET9,X,Y;SOUND55,100;PAUSE10;G.b
160GOS.a;AS.TOR,9;Z=0;G.b
170GOS.a;AS.WEE,9;Z=0;G.b
180GOS.a;AS.CON,9;Z=0;G.b
190GOS.a;AS.WES,9;Z=0;G.b
200GOS.a;AS.AND,9;Z=0;G.b
210GOS.a;AS.OR,9;Z=0;G.b
220GOS.a;AS.INV,9;Z=0;G.b
230GOS.a;AS.LYN,9;Z=0;G.b
240GOS.a;AS.DIO,9;Z=0;G.b
250GOS.a;AS.ELC,9;Z=0;G.b
260GOS.a;AS.QTR,9;Z=0;G.b
270GOS.a;AS.PYL,9;Z=1;G.b
280GOS.a;AS.PYL,9;Z=3;G.b
290GOS.a;AS.PYL,9;Z=5;G.b
301PAINT X,Y,1;G.b
302PAINT X,Y,2;G.b
304PAINT X,Y,4;G.b
305PAINT X,Y,5;G.b
306PAINT X,Y,6;G.b
307PAINT X,Y,7;G.b
308PAINT X,Y,8;G.b
309PAINT X,Y,9;G.b
310PAINT X,Y,0;G.b
311PAINT X,Y,10;G.b
312PAINT X,Y,11;G.b
313PAINT X,Y,12;G.b
314PAINT X,Y,  ;G.b
319aDE.TOR;DE.WEE;DE.CON;DE.WES;DE.AND;DE.OR
320DE.INV;DE.LYN;DE.DIO;DE.ELC;DE.QTR;DE.PYL;R.
350TURN9;SOUND75,50;PAUSE10;G.b
400TXMOD;P.$U;COS1
410IN."SAVEN J/N"$T;IF$T="J" *SAVE"TEKBEST"8200 9500
490P."EINDE";E.
700PLAY D'8;?#E1=#FF;IN.$T;P.$30$10;G.80
800PLOT4,X,Y;SOUND 55,100;PAUSE10;Z=2;G.b
810PLOT6,X,Y;SOUND 55,100;PAUSE10;Z=1;G.b
820R=X;SOUND 55,100;PAUSE10;Z=4;G.b
830CI.1,R,Y,ABS(X-R);SOUND 55,100;PAUSE10;Z=3;G.b
840R=X;S=Y;SOUND 55,100;PAUSE10;Z=6;G.b
850T=ABS(X-R);W=ABS(Y-S);CU.2,0,R,S,T,W;SOUND 55,100;PAUSE10
860Z=5;G.b
1000REM HEB JE GEEN JOYSTICK; GEEN NOOD!!
1001REM REGEL 88 WEGHALEN
1002REM        91  Y=Y+1;G.b
1003REM        92  Y=Y-1;G.b
1004REM        93  X=X+1;G.b
1005REM        94  X=X-1;G.b
1006REM        87  ATK.(G,T,B,H,F)(100,91,92,93,94)
1007REM EN NU DOEN DE TOETSEN G,T,B,H EN F HET.
```

Dit programma, dat is ontstaan uit de kreet:
"PROGRAMMEREN? EEN KIND KAN DE WAS DOEN!!!!"
is meer een grap, dan functioneel.
Het is een plaatje van een wasmachine met atributen zoals zeepbakjes, kraan, stopkontakt enz. Deze atributen staan ook links in een rij op het scherm. Hierbij staat een pijl, die men met de joystick naar boven en beneden kan verplaatsen. Drukt men op de vuurknop, dan wordt het symbool naast de pijl overgenomen en op een rij rechts op het scherm neergezet. Tevens wordt hierdoor een commando bij de wasmachine uitgevoerd (b.v. stekker in de wandkontaktdoos). Als men de juiste commando's uit laat voeren, wordt de trui gewassen. Doe je het niet goed, dan ....
Vergeet b.v. de temperatuur in te stellen, dan is de trui gekrompen als je hem uit de wasmachine haalt. Als je met dit programma goed geoefend hebt mag je misschien "moeder de vrouw" een helpen.

Opmerking: Heeft men geen josbox dan:
regel 90 GRMOD wordt CLEAR 4
regel 100 weghalen
regel 690 TXMOD wordt CLEAR 0

```
>P>L.
    1REM NODIG gagsrom EN josboxJOYSTICK VOLGENS CLUBAANSLUITING
   10P.$12"****WASMACHINE   BY G.T.H.****"'''
   20P."ONDER HET MOM VAN"'"""PROGRAMEREN? EEN KIND KAN DE"'
   30P." WAS DOEN!!""""'"DIT PROGRAMMA."'"-PIJL BEWEGEN MET JOYSTIC"
   40P."K."'"-MET DE VUURKNOP COMMANDO GEVEN."
   50P."-MET RETURN VERDER."''
   60DIMG(1)
   70IN.$G;IF$G=""G.90
   80G.70
   90?#B401=0;?#E1=0;GRMOD;B=165;A=0;C=0;Q=0;Z=0;X=0;Y=0;R=1
  100P.'"  ***WASMACHINE***BY G.T.H.***"''''''''''"             "
  110?#B402=0
  120 BASE((TOP/256)+1)
  130CR. KRAAN,#38#3F#3F#3F,1,1,7,7,12#FC#FC#FC,12,0#C0#C0
  140CR.KRN,16,16,31,0,0,0,4,4,0,0#FC,0,0,0#40#40
  150CR. STOP,#80#84#BC#FF#3C,4,0,0,1#3F#3F#BF#3F#3F,1,0
  160CR.STK,0,4,#3F,0#3F,4,0,0,0#40#C0#40#C0#40,0,0
  170CR. KLOK,6,8,16,17,17,9,5,2#C0#20,16#D0,16#20#40#80
  180CR. SCHAK,7#E5#E5#E0#E5#E7#E5,7#48#48#5C,0#55#75#57#75
  190CR.SCHK,0,0#40#40#40#40,0,0,0,0,0,0,0,0,0
  200CR.ZEEP,#38#7C#7C#7C#7C#FE#FE#FE,#38#7C#7C#7C#7C#FE#FE#FE
  210CR.ZP,0,16#38#38#38#38#7C,0,0,16#38#38#38#38#7C,0
  220CR./P:5 #AA,#88,#AA,#88,#AA,#88,#AA,#88
  230CR.KLE,#1F#1F#1F#FF#FF#FC#FC#FC#F8#F8#F8#FF#FF#3F#3F#3F
  240CR.REN,#1F#1F#1F#1F#1F#1F#1F#1F#F8#F8#F8#F8#F8#F8#F8#F8
  250CR.TEMP,14,14,14,4,4,4,4,4,0,0,14,8,8#EE#A0#E0
  260CN /P:15 #AA,#AA,#AA,#AA,#AA,#AA,#AA,#AA
  270CR./P:28 #80,#E0,#E1,#F3,#CF,#87,7,1
  280CR.PIJL,#18#30#60#FF#FF#60#30#18,0,0,0#FF#FF,0,0,0
```

```
290BO.5,5;BO.10,10;BO.79,50;FORI=20 TO 25 S.5;CI.1,128,96,I;N.
300MOVE80,130;DRAW175,130;MOVE98,130;DRAW98,140
310MOVE179,132;DRAW179,100;DRAW176,100
320MOVE142,152;DRAW142,142;MOVE144,152;DRAW144,142
330MOVE60,10;DRAW60,168;MOVE195,10;DRAW195,168
340MOVE60,55;DRAW195,55;MOVE10,168;DRAW245,168
350SET:ZEEP,85,149;SET:STOP,179,140;SET:KRAAN,140,160
360SET:SCHAK,81,139
370PA.100,133,15
380PA.85,127;PA.160,96
390PA.7,38,5;PA.247,153,5
400PA.65,50,28;PA.190,50,28
410SET:KRAAN,15,165;SET:PIJL,43,165;SET:STOP,15,141
420SET:KLOK,15,117;SET:SCHAK,15,93;SET:ZEEP,15,69
430SET:KLE,15,49;SET:REN,15,41;SET:TEMP,15,20
440NOSNOW;X=165
450J.A,B,C
460IFC<>0 G.500
470IFB<20 B=20
480IFB>165 B=165
490CARRY:PIJL,43,B;G.450
500IFB<38G.t
510IFB<60G.k
520IFB<82G.z
530IFB<104G.s
540IFB<126G.c
550IFB<148G.w
560SET:KRAAN,210,X;SET:KRN,140,160;Z=Z+1;IFZ=3 Z=Z-2;G.e
570G.e
580tSET:TEMP,210,X;R=R-1;G.e
590kIFR=2 G.r
600R=R+1
610 SET:KLE,210,X;X=X-8;SET:REN,210,X;SET:KLE,121,85;G.e
620rSET:KLE,210,X;SET:KLE,121,85;G.e
630zSET:ZEEP,210,X;SET:ZP,85,149;G.e
640sSET:SCHAK,210,X;SET:SCHK,81,139;GOS.a;G.e
650cSET:KLOK,210,X;Q=Q+50;G.e
660wSET:STOP,210,X;SET:STK,179,140;Z=Z+1;IFZ=3 Z=Z-2
670eX=X-12;IF X<10 G.690
680PAU.20;G.450
690?#B401=#0;TXMOD;G.90
700aIFZ<>2 SET:SCHK,81,139;R.
710PLOT4,121,79;PLOT2,14,0;PAUSE40
720PLOT4,117,81;PLOT2,22,0;PAUSE40
730PLOT4,115,83;PLOT2,26,0;PAUSE40
740PLOT4,113,85;PLOT2,30,0;PAUSE40
750PLOT4,112,87;PLOT2,32,0;PAUSE40
760FORY=0TOQ
770PLOT4,111,89;PLOT1,34,0;PAUSE10;PLOT2,-34,0;PAUSE10
780NEXTY;PAUSE40
790PLOT4,112,87;PLOT2,32,0;PAUSE40
800PLOT4,113,85;PLOT2,30,0;PAUSE40
810PLOT4,115,83;PLOT2,26,0;PAUSE40
820PLOT4,117,81;PLOT2,22,0;PAUSE40
830PLOT4,121,79;PLOT2,14,0;PAUSE40
840SET:SCHK,81,139;R.
850E.
```

Omdat het toch wel makkelijk is, als je zelf E-proms kunt programeren (of copieren), heb ik een E-programeerapparaatje voor de Acorn Atom ontwikkeld. Zoals een goed hobbyist betaamd stelde ik de eis dat hij goedkoop moest zijn. Dit is mij dacht ik wel gelukt.

De hardware.

Het eenvoudigst zou het zijn om de I/O-poort van de ATOM direct te koppelen aan de adres- data- en stuurlijnen van de te programeren E-prom. De I/O-poort heeft echter maar 16 parallelaansluitingen en de E-proms hebben er 22 (ex. voeding). Maar geen nood, een teller 4040 gebruikt als adresmedia bracht de oplossing. Verder bestaat de schakeling uit een stuurtransistor die de 25V schakelt en nog enige buffer-inverters, twee ledjes, waarvan een aangeeft of de voedingsspanning aanwezig is en de andere of de programeerspanning op de E-prom staat. Rest nog de schakelaar, waarmee men de keuze kan maken tussen een 2716 of een 2732 E-prom. Het bijbehorende progamma checkt dit af. Met behulp van een tussenvoetje (zie aansluitgegevens) kan je ook de 2532 programeren.

De software.

Met het bijbehorende progamma kun je verschillende dingen doen. De keuze wordt met behulp van een "menu" bepaald. Voordat dit menu verschijnt dient eerst het begin- en eindadres (met # voor hexadecimale getallen) van het geheugen dat bewerkt moet worden.

De mogelijkheden zijn:
- inladen vanuit E-prom
- handmatig ingeven
- programeren van de E-proms
- kontroleren (vergelijken van het geheugenen E-proms.
- hex-listen
- eindigen.

Hierna wordt gevraagd of men met een 2716 of een 2732 E-prom wilt werken.
N.B. Het progamma heeft de P-charm nodig.

Het schema en het progamma staan op de volgende bladzijden.

En nu de bouw.

Ten eerste hebben we een enkelzijdig printplaatje nodig. De layout staat op ware grootte afgedrukt op de pagina hiernaast. Dit is dus voor "de hardware-doe-het-zelver" geen probleem meer.

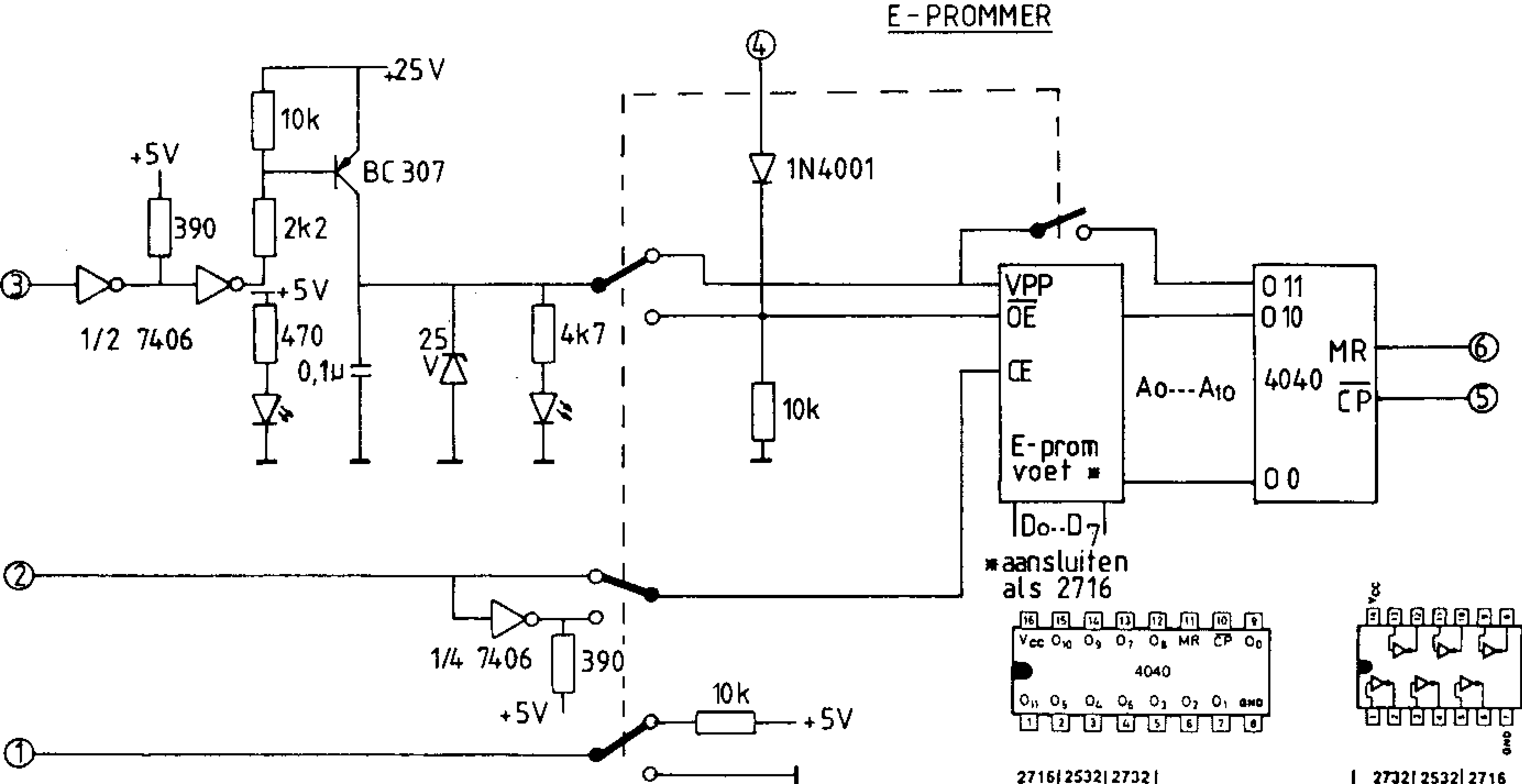
De onderdelen die we nodig hebben zijn:

R1- 390 Ohm	D1- zener 25 V	1 schakelaar 4x wissel
R2- 390 Ohm	D2- 1N4001	1 IC-voet 24 pins (evt. progameervoet)
R3- 10 kOhm	D3- led rood	2 beugels volgens fig.
R4- 2,2 kOhm	D4- led groen	1 64-polige connector female
R5- 10 kOhm	C1- 0,1 uF	4 schroefjes + 4 moertjes M3
R6- 4,7 kOhm	T1- BC 307	1 printplaatje
R7- 10 kOhm	IC1 SN7406	2 soldeerpennetjes (t.b.v. 25V)
R8- 470 Ohm	IC2 4040B	

- Als de print geetst en geboord is brengen we eerst de elf draadbruggetjes aan, en soldeer deze vast. (zie pijltjes)
- Dan plaatsen we de E-promvoet, de schakelaar en de soldeerpennetjes. Soldeer deze vast.
- Nu solderen we resp. de weerstanden, condensator, dioden (let op de richting), led's en de transistor er op.
- Soldeer daarna de IC-'s erop en klaar is de print.

Hoe sluiten we hem aan op de ATOM, zodat je goed bij de E-promvoet en de schakelaar kunt komen?
Met twee metalen beugeltjes van 8x30 mm, omgezet op 10 mm, (zie fig.) monteren we een 64-polige connector aan het printje. De voeding en de massa van het printje sluiten we met draadjes aan op resp. punt 1 en 32 van de B-rij van de connector. De overige gebruikte punten, van het printje, verbinden we recht (3 op 3, 4 op 4 enz.) met de A-rij van de connector. Als men nu de connector achter in de ATOM prikt en 25 V aansluit (let op + en -) is de EPROMMER klaar voor gebruik.


fig.



2716	2532	2732	pin		pin	2732	2532	2716
A7	A7	A7	1	VOET	24	VCC	VCC	VCC
A6	A6	A6	2		23	A8	A8	A8
A5	A5	A5	3		22	A9	A9	A9
A4	A4	A4	4		21	A11	VPP	VPP
A3	A3	A3	5		20	OE/VPP	CE	OE
A2	A2	A2	6		19	A10	A10	A10
A1	A1	A1	7		18	CE	A11	CE
A0	A0	A0	8		17	D7	D7	D7
D0	D0	D0	9		16	D6	D6	D6
D1	D1	D1	10		15	D5	D5	D5
D2	D2	D2	11		14	D4	D4	D4
GND	GND	GND	12		13	D3	D3	D3

D_0--D_7 = poort B_{0-7}

ⓧ = poort A_x

32
25 v voed.
STAND
PGM
ST25v
LEES
Ø
MR
D7 - - - - - - - - D0
+
G.T.H.
E-PROMMER
R1
IC1
R2
R3
R4
T1
C1
D1
R7
D2
E-PROM
IC2
R8
R6
R5
D3
D4

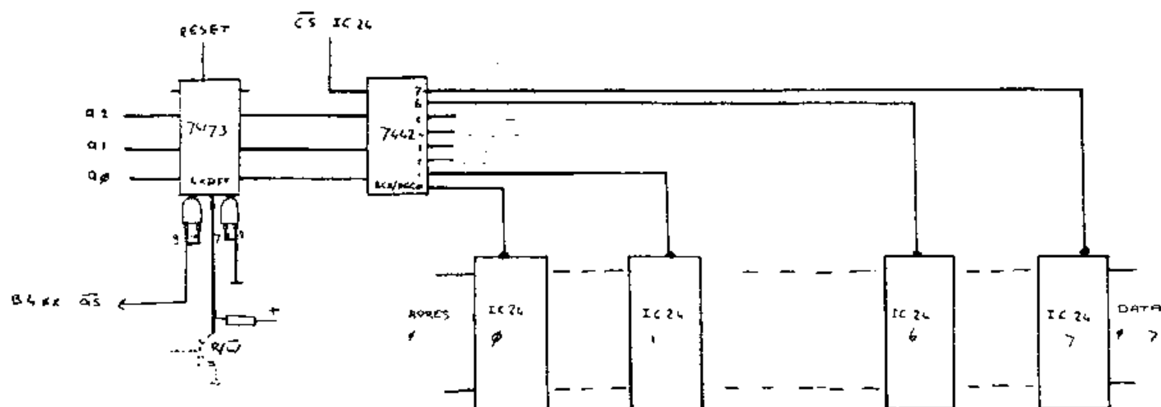
1
32
G.T.H.
E-PROMMER
+
25
v

```
  5REM P-CHARM NODIG VOOR HEXLISTEN.
 10P.$12,"*PROGRAMEREN E-PROMS BY G.T.H.**"'$15
 15?#B803=#FD;?#B801=#48
 20IN."BEGINADRES ",B
 30IN."EINDADRES  ",E
 40xS=B;P.'
 50P."INLEZEN =        <I>",'
 55P."HANDINVOER=      <H>",'
 60P."PROGRAMEREN =    <P>",'
 70P."KONTROLEREN =    <K>",'
 75P."HEX-LIST=        <X>",'
 80P."NIEUWE ADRESSEN=<N>",'
 90P."EINDIGEN =       <E>",'
100IN.,A;IF A=E P.$12,"EINDE";E.
102IFA=H G.h
103IFA=X P.$14;HEX B
105IFA=N G.10
120P."DE SCHAKELAAR STAAT OP "
130IF?#B801&#02=0P."2732"'$7;Z=1;G.150
140P."2716",$7,';Z=0
150IN."STAAT DEZE GOED J/N"C
160IFC=N P.$11,$11;G.120
170IFA=P GOS.p
180IFA=I ORA=K GOS.i
190rF.Q=1TO11;P.$11;N.Q;G.x
200p?#B802=#FF
220?#B801=#50
230?#B800=?S
240?#B801=#14
250WAIT;WAIT;WAIT
260?#B801=#30
280S=S+1;IFS<=E;G.230
285?#B801=#48
290P.$7;R.
300i?#B802=#00
310?#B801=#48
320?#B801=#08;IF Z=1THEN?#B801=#0C
330IFA=K GOS.k
340IFA=I R=?#B800
345?S=R
350?#B801=#28
360S=S+1;IFS<=E;G.320
370P.$7;R.
400kIF?#B800=?S;G.420
410P."FOUT OP ADRES ",&S,',$7
420R.
430hP.$12"VERLATEN MET LETTER >F"';@=1
440P.&B" "
450FOR G=0 TO 3;INK.C
460C=C-48;IF C>10 C=C-7;IF C>15 @=8;G.10
470P.&C;D=C*16
480INK.C;C=C-48;IF C>10 C=C-7
490P.&C;?B=(C+D);B=B+1;P." "
500NEXT G
510P.'&B" ";G.450
520E.
```

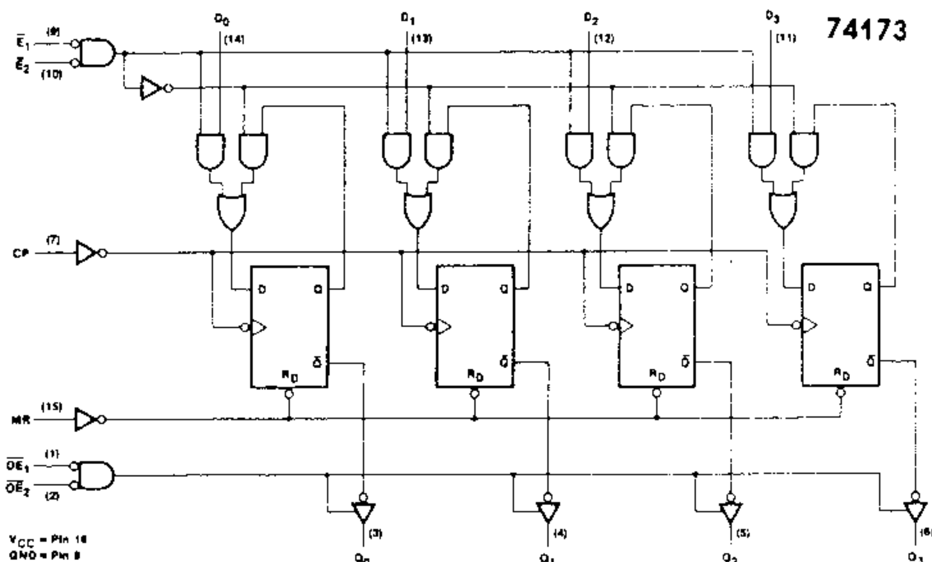
SCHAKELKAART 8xIC 24

De werking:

De werking van deze schakelkaart berust op, het bij velen reeds bekende principe van, het sturen van de chip select ($\overline{cs}$) aansluiting van de E-prom's. Omdat het adresgebied (van IC 24 van ≠AØØØ tot ≠BØØØ) voor alle te schakelen IC-s het zelfde zijn mag er slechts één E-prom aangeschakeld zijn. Dit houdt in dat van één IC de chip select lijn laag mag worden en de andere hoog moeten blijven. Hoe is dit te realiseren?



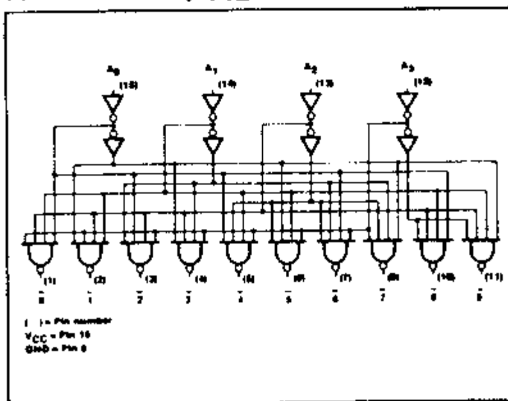
Ik ben er achter gekomen dat de adresselectie van IC 49 slechts voor de helft wordt benut voor de VIA (≠BØØØ t/m ≠B3FF) en de PPI (≠B8ØØ t/m ≠BBFF). Hier heb ik dankbaar gebruik van gemaakt. Het adresselect signaal ($\overline{as}$) van poot 5 van IC 49 gebruik ik om Flip-Flop's (soort geheugen) te enable-en. Het IC 74173 is hiervoor een prima IC.



Van de vier D- Flip-Flop's die hierin zitten worden er drie gebruikt om de laagste adressen in te bewaren. De adreslijnen AØ, A1, en A2 gebruik ik als schakelmedia. De ingangen van de D-Flip-Flop's worden slechts geopend als er wordt "geroerd" in het adresbereik ≠B4ØØ t/m B7FF. De aangeboden data wordt ingeklokt door het Read signaal (r/$\overline{w}$) of als iemand dat wil,

door het Write signaal (transistor toepassen voor invertering). Omdat ik wil dat er een bepaalde E-prom (de P-charm) voorgeschakeld staat na inschakeling of een Break wordt de ATOM-reset aangesloten op de reset van de 74173 (mr). Als men nu intoetst ?≠B4Ø7=Ø worden de Flip-Flop's geset. Met drie Flip-Flop's hebben we acht mogelijkheden. Daartoe dienen de uitgangen van de Flip-Flop's gedecodeerd te worden.
Hiervoor is het IC 7442 ontworpen.

LOGIC DIAGRAM 7442

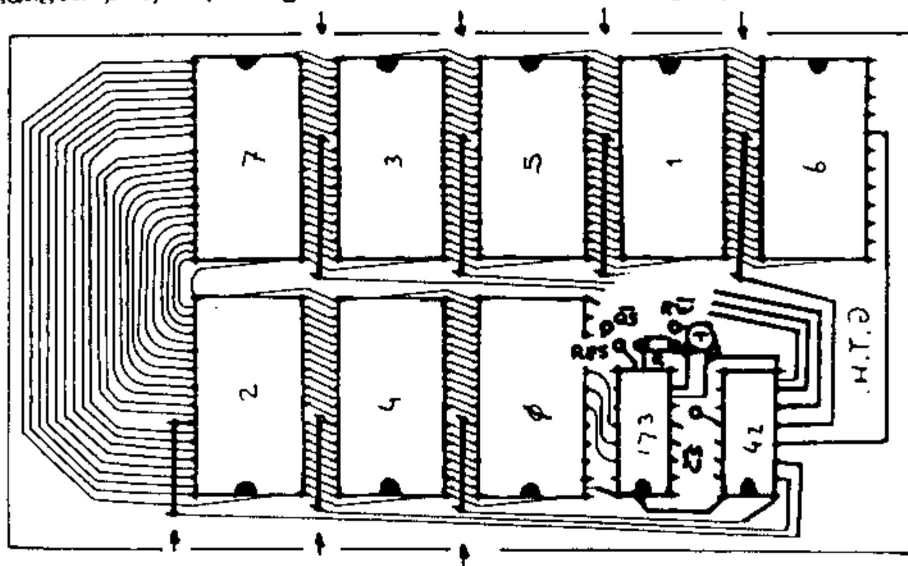


FUNCTION TABLE

A_3	A_2	A_1	A_0	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{8}$	$\bar{9}$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	H	H	L	H	H	H	H	H	H	H
L	L	H	H	H	H	H	L	H	H	H	H	H	H
L	H	L	L	H	H	H	H	L	H	H	H	H	H
L	H	L	H	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
L	H	H	H	H	H	H	H	H	H	H	L	H	H
H	L	L	L	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
H	L	H	L	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
H	H	L	L	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	H	L	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = HIGH voltage levels
L = LOW voltage levels

Als men hiervan drie van de vier ingangen gebruikt en de vierde ingang laag houdt heeft men op de uitgangen Ø t/m 7 één uitgang laag. Deze sluiten we aan op de $\overline{cs}$ van de E-prom's. Om de E-proms alleen te laten werken in het adresgebied van ≠ AØØØ tot BØØØ wordt de $\overline{cs}$-lijn van IC 24 aangesloten op de vierde ingang, waardoor er slechts één van de uitgangen Ø t/m 7 laag wordt als de vierde ingang laag is.



Bouwbeschrijving:

Onderdelen: 1- SN7442
1- SN74173
1- weerstand 10 k-Ohm
9- 24 pins IC-voetjes
1- printplaat (layout is beschikbaar)
en enige dosis vrije tijd.

Leg eerst de 7 draadbruggetjes (zie pijltjes).
Monteer dan de weerstand en vergeet de transistor.
Soldeer de IC-voeten Ø t/m 7
Soldeer de SN 74173 en de SN 7442.

Soldeer stugge draadjes, die in de componentenzijde van het extra soldeervoetje passen in het IC-voetje (met ronde pootjes) Poot 20 vrijhouden. Knip de draadjes op een lengte van 1 cm. af. Soldeer deze onder IC 24/0 van de kaart. Let op de juiste telling.
Sluit op poot 20 van de extra IC-voet de $\overline{cs}$-draad aan.
Soldeer draadjes aan $\overline{as}$, res, r/$\overline{w}$(op gat van T. bij R.) en aan de onderzijde $\overline{cs}$.
Sluit de RES draad aan op de resetlijn van de ATOM (poot 35 van IC 25)
Sluit de $\overline{as}$ draad aan op poot 5 van IC 49 (uitbuigen)
Sluit de r/$\overline{w}$ draad aan op de r/$\overline{w}$ lijn van de ATOM

Prik de E-proms in een willekeurige plaats, maar zor er voor dat de P-charm op positie Ø zit.

Proberen maar.

Na inschakelen of na een break staat IC Ø voor.

Omschakelen: ?≠ B4Ø1 = Ø IC 1
?≠ B4Ø2 = Ø IC 2
enz.
?≠ B4Ø7 = Ø IC 7
?≠ B4ØØ = Ø IC Ø

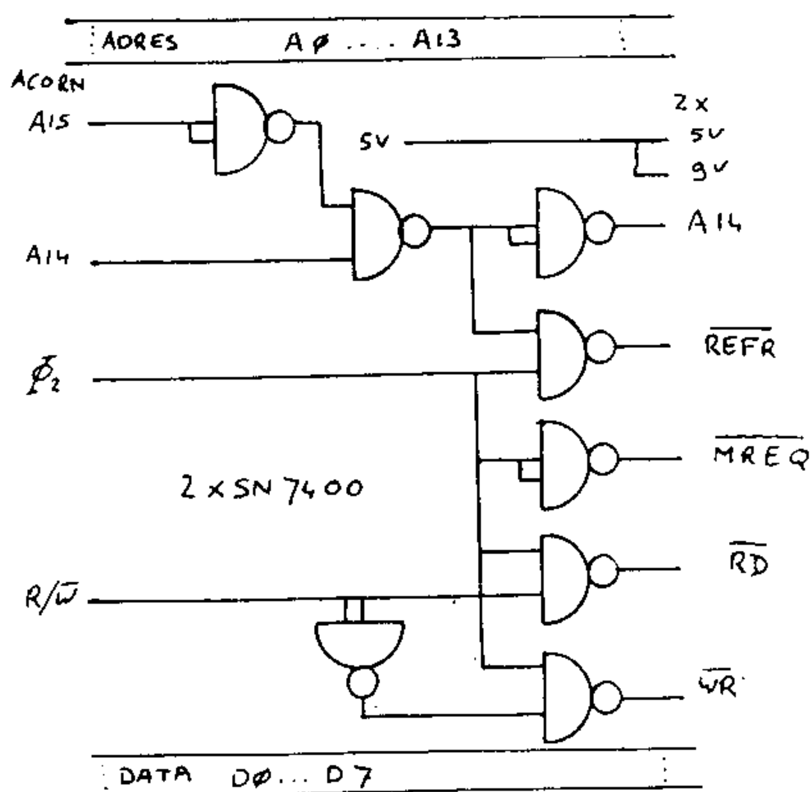
Automatisch omschakelen gaat ook. Hiervoor is echter een programma nodig. Dit programma dient geladen en gerund te worden, voordat men andere programma's die automatisch geschakeld moeten worden,kan runnen.

De listing van het schakelprogramma is hierbij afgedrukt.

```
>>L.
   5REM SCHAKELPROGRAMMA VOOR SCHAKELKAART 8*IC 24
  10P.$12"ACORN ATOM SWITCH"''
  30P.$21;DIM BB(4);F.I=1TO2;P=#9ED8
  40[
  50:BB1;LDX@0;PLA;PLA;STA0
  60       CMP@94;BNEBB3
  70       LDX#5;CPX#88;BEQBB2
  80       STX#88;LDX@0;STX#89
  90:BB2;LDX#89;INX
 100       STA#B401;CPX@2;BEQBB4
 110       STA#B402;CPX@3;BEQBB4
 120       STA#B403;CPX@4;BEQBB4
 121       STA#B404;CPX@5;BEQBB4
 122       STA#B405;CPX@6;BEQBB4
 123       STA#B406;CPX@7;BEQBB4
 124       STA#B407;CPX@8;BEQBB4
 130:BB3;STA#B400;CPX@1;BEQBB4
 140       LDX@0;STX#89;JMP#ABD5
 150:BB4;STX#89;LDA#A001;CMP@#BF;BNEBB2
 160       JMP#A002
 170]N.I;?#9ED7=#9E;P.$6;E.
```

GOEDKOPE 16 k-RAM UITBREIDING. met het vierkante ZX-Sinclair blok.

Nu het enthousiasme van de ZX-81 bezitter begint over te slaan naar teleurstellingen komen er in de tweedehands-handel steeds meer geheugen-uitbreidingsblokken van de ZX-81 beschikbaar. Ook ik kwam voordelig aan zo'n blok. Na enig gevogel lukte het mij om zo'n blok op de ACORN-ATOM aan te sluiten. Hiervoor zijn slechts twee IC-tjes nodig. Het schema spreekt voor zich. Voor de uitvoering hiervan is "eilandjes print" met geëtste connector zeer geschikt. Let wel op de steekafstand van de connector. Hier moet het geheugenblok op passen. Met een extra connector kan het geheugen aan de ATOM gekoppeld worden.(zie afbeelding).



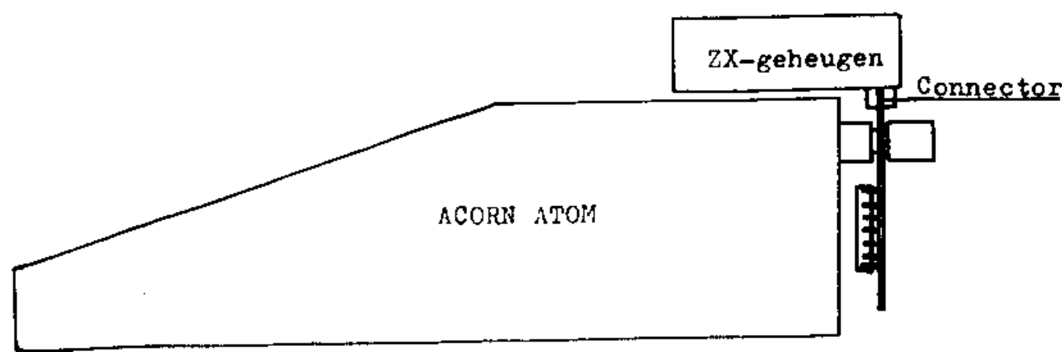
AANSLUITING ZX-GEHEUGEN OP DE ACORN ATOM

ONTWERP BY G.T.H.

TOP

D7	5V
*	9V
KEY	KEY
DØ	ØV
D1	*
D2	*
D6	AØ
D5	A1
D3	A2
D4	A3
*	*
*	A14
*	A13
MREQ	A12
*	A11
RD	A10
WR	A9
*	A8
*	A7
*	A6
*	A5
*	A4
RFS	*

* = N.C.



SOFTWARE: extended autoshow

We kennen nog wel onze oude vertrouwde AUTOSHOW.BAS. Dit programma liet alle plaatjes in vogelvlucht zien,die op een schijf staan met de extensie .PIC.
Deze 'extended' versie doet niets anders, doch de plaatjes vloeien in elkaar over m.b.v. de overbekende FLOW routine en men kan de plaatjes dumpen op papier door de letter -D- in te drukken op het moment dat het plaatje op het beeldscherm staat. De dumproutine is SDUMP uit de RXBOX. Voor de rest is men GDOS 1.5 en P-Charme nodig.

```
 10 PROGRAM AUTOSHOW.EXT
 20
 30 PROC ASSEMBLE
 40 P.$21;FOR I=1 TO 2
 50 P=#2800;[
 60:SS0
 70 LDX@0;JSR SS2
 80 LDX@1;JSR SS2
 90 LDX@2;JSR SS2
100 RTS
110:SS2 STX Z;STX Z+2
120 LDX@#80;STX Z+1
130 LDX #80;STX Z+3
140:SS3 LDY@0
150 LDA(Z+2),Y;STA(Z),Y
160 INC Z;INC Z+2;BNE P+6
170 INC Z+1;INC Z+3
180 INC Z;INC Z+2;BNE P+6
190 INC Z+1;INC Z+3
200 INC Z;INC Z+2;BNE P+6
210 INC Z+1;INC Z+3
220 LDA Z+1;CMP@#98;BMI SS3
230 JSR#FB8A;RTS
240];NEXT;P.$6
250
260 PEND
270
280 Z=#70
290 DIM SS5
300 FORI=0TO5;SSI=-1;N.
310 ASSEMBLE
320 *NOMON
330 CLEAR4
340 FOR I=0 TO 1
350  *OSCLI "DRIVE",I,$13
360  *DIR
370  N=?#200F
380  FOR J=N-1 TO 0 STEP -1
390   ?(#201F+J*16)=13
400   XIF INSTR($(#2010+J*16),".PIC")
410    T=?(#2506+J*8)
420    S=?(#2507+J*8)
430    T=T*16;T=T+(S/16)
```

```
440     S=S%16
450     IF S=0;S=16;T=T-1
460     @=0
470     *OSCLI "RSECT ",&T," ",&S," "," 18 6800",$13
480     PAUSE 60
490     ?#80=#68;LINK#2800
500    ELSE
510   NEXT J
520   PAUSE 60
530 NEXT I
540
550 *MON
560 PAUSE 90
570 P.$12
580 END
```

SOFTWARE: Disc>disc

Het programma disc>disc is een GDOS utility en afgeleid van de eerder gepubliceerde utilities ACORN->GDOS en SQUEEZE.Het programma biedt de keuze tussen het in z'n geheel copieren van een schijf, of het copieren van enkele willekeurige files van de ene naar een andere schijf. In het laatste geval heeft U tevens de mogelijkheid om de te copieren files te renamen.
Bij gebruik van 1 drive geeft het programma steeds aan welke schijf (source- of target-disc) in de drive gestopt moet worden; bij gebruik van twee drives verloopt het copieer-proces volledig automatisch.
Het programma gebruikt als copieerbuffer het geheugengebied van #2900 tot #8000; dit is eventueel te veranderen door aanpassing van de variabelen A en K. Na het copieren eindigt het programma met het KEEP-en van de directory van de target-disc en toont deze tevens op het beeldscherm.

Dick Protzman

PRINTERWARE: Dumpprogramma voor GP50

Onderstaand programma is bedoeld om een grafische dump van een plaatje in CLEAR 4-mode te maken op een Seikosha GP-50 printer.Uit de listing blijkt dat het te dumpen plaatje op #5000 geladen moet worden; de object code van de dumproutine staat op #800. Daar de source in SALFAA geschreven is kan dit makkelijk terug gevonden worden; zodat de code makkelijk naar een andere geheugenlocatie verplaatst kan worden.
Verder gebruikt SALFAA het gebied #4000 tot #5000 als ruimte voor de SYMBOL table.

```
 10 \\ SCREEN DUMP \\
 20 \\ VERSION 0.0 \\
 30
 40 PASS 2;GOS.a
 50 PASS 1;GOS.a
 60
 70 P=P&#0000FFFF;@=0
 80 P.$12'''"CODE VAN "&start
 90 P." TOT "&P'
100 P.$7$7;LINK #FFE3
110 CLEAR4;COPY#5000,#6800,#8000
120\INV
130 LINK #0800
140 P.$12'''"END"';END
150
160aASM-BEGIN
170 .OPTION #40
180 .TABLE #4000,#4FFE
190
200 :zp         =#B0
210 :start'assembly=#07FF
220 :pr'drive       =#FF10
225 :pr'drv         =#FEFB
230 :new'line       =10
240
250 .CODE start'assembly
260 .RAM  start'assembly
270
280:NEP'start NOP
290:start JSR lbe'13;BEQ lbe'3
310:lbe'13 LDA@2;JSR pr'drv
320 LDA@27;JSR pr'drive
330 LDA@48;JSR pr'drive
```

```
340:lbe'0 LDA@16;JSR pr'drive
342 LDA@0;JSR pr'drive
344 LDA@4;JSR pr'drive
345 LDA@27;JSR pr'drive
350 LDA@71;JSR pr'drive
360 LDA@1;JSR pr'drive
370 LDA@0;JSR pr'drive
380 RTS
390:lbe'3 LDA@#80;STA zp+1
400:lbe'10 LDA@0;STA zp
410 STA zp+2
420:lbe'11 LDY@0;STY zp+5
430:lbe'12 LDA@8;STA zp+3
440:lbe'4 LDA zp+3;STA zp+4
450 LDA(zp),Y;LDX zp+2
460 AND lbe'20,X;BEQ lbe'5
470 LDA@#80
480:lbe'5 DEC zp+4;BEQ lbe'9
490 LSRA;JMP lbe'5
500:lbe'9 CLC;ADC zp+5;STA zp+5
510 DEC zp+3;TYA;CLC;ADC@#20
520 TAY;BNE lbe'4
530 LDA zp+5;EOR@#FF
540 CMP@#80;BCC lbe'14
550 PHA;LDA@8;EOR#B002;STA#B002
560 PLA;JSR pr'drive
570 LDA@8;EOR#B002;STA#B002;JMP lbe'15
580:lbe'14 JSR pr'drive
590:lbe'15 INC zp+2;LDA zp+2
600 CMP@8;BNE lbe'11
610 INC zp;LDA zp;CMP@#20;BEQ lbe'6
620 LDY@0;STY zp+2;STY zp+5;BEQ lbe'12
630:lbe'6 LDA@ new'line;JSR pr'drive
640 INC zp+1;LDA zp+1
650 CMP@#98;BEQ lbe'7
660 JSR lbe'0;JMP lbe'10
670:lbe'7 LDA@27;JSR pr'drive
680 LDA@50;JSR pr'drive
690 LDA@3;JSR pr'drv
700 RTS
710:lbe'20
720.BYTE #80,#40,#20,#10,8,4,2,1
730.END
740 RETURN
```

Software

SOFTWARE: Backup disc-to-tape

De ACORN-DOS bezitters onder U kennen ongetwijfeld het programma BACKUP van Ronald Boers.De GDOS bezitters onder ons zullen dan ook geconstateerd hebben dat eerder genoemd programma in het geheel niet bruikbaar is voor GDOS. Welnu, daarin komt nu verandering met het hierna beschreven programma.

Algemeen:

Een nadere bestudering van de opbouw van een GDOS directory leert ons dat hierin maximaal plaats is voor 78 entries met een maximale filenaam lengte van 15 karakters. Dat beteken: dus dat over het algemeen de directory niet op 1 normaal Atom scherm past zoals dat bij BACKUP het geval was.Derhalve is de opzet van het nieuwe Backup programma dan ook totaal anders dan die van BACKUP.

Opbouw:

Het programma begint met het opvragen van een aantal gegevens omtrent uw DOS en COS; namelijk:

- backup maken van welke drive [0/1/2/3]
- aantal tracks op de schijf [40/80]
- gewenste Baudrate voor COS [300/1200]

Vervolgens wordt de desbetreffende directory .gelezen waarna gevraagd wordt of de gehele schijf naar tape moet of dat U een selectie wilt maken uit de directory.
Bij het backuppen van de gehele schijf hoeft U nu alleen nog maar de cassetterecorder aan te zetten en op de spatiebalk te drukken waarna de gehele schijf file voor file naar tape gestuurd wordt met de gewenste Baudrate

Bij het selectief backuppen van een schijf worden alle filenamen een voor een op het scherm getoond, waarbij U dan uit een aantal opties kunt kiezen. Deze zijn:

- Yes : plaatst file in de backup-queue
- 'o : file komt niet in backup-queue
- Info : geeft *INFO van de desbetreffende file
- Rename : biedt mogelijkheid filenaam te wijzigen

Wanneer een file in de backup-queue geplaatst moet worden waarvan de filenaam langer is dan 13 karakters dan worden de eerste 13 karakters getoond en vervolgens wordt gevraagd of men hiermee accoord gaat (Accept) of niet (Reject).
In het eerste geval wordt de file met de tot 13 karakters ingekorte filenaam in de backup-queue geplaatst; in het tweede geval kan men via de optie Rename zelf een andere filenaam invullen waarna de file alsnog in de backup-queue geplaatst wordt.

Nadat de gehele directory doorlopen is moet men de cassette recorder op opname zetten en vervolgens op de spatiebalk drukken waarna de gewenste files naar tape gestuurd worden.

Het beantwoorden van alle vragen, alsmede het kiezen uit de mogelijke opties, gebeurt door het intypen van het eerste karakter van de aangegeven mogelijkheden (staan tussen vierkante haken).

Uitvoering:

Tijdenshet backup proces zijn op het scherm de volgende zaken te zien:

- Gekozen Baudrate
- Filenaam op schijf
- Filenaam op cassette
- Start-, Eind- en Executie adres plus de Lengte in blokken

Na afloop van het backup proces wordt er enig geluid geproduceerd ten teken dat het programma klaar is.

Implementatie:

Voor een goede werking van het programma moeten de volgende zake in de Atom aanwezig zijn:

- GDOS 1.5
- PCHARME
- JOSBOX
- Goede schakelsoft

Deze laatste twee zijn noodzakelijk voor de FCOS en SCOS routines. Eventueel, voor niet JOSBOX bezitters, kan er ook gebruik gemaakt worden van de COS routines uit PCHARME. Dan moeten de volgende regels in de procedure FILE-SAVE:

```
IF C=1;FCOS
IF C=0;SCOS
```

vervangen worden door: COS C

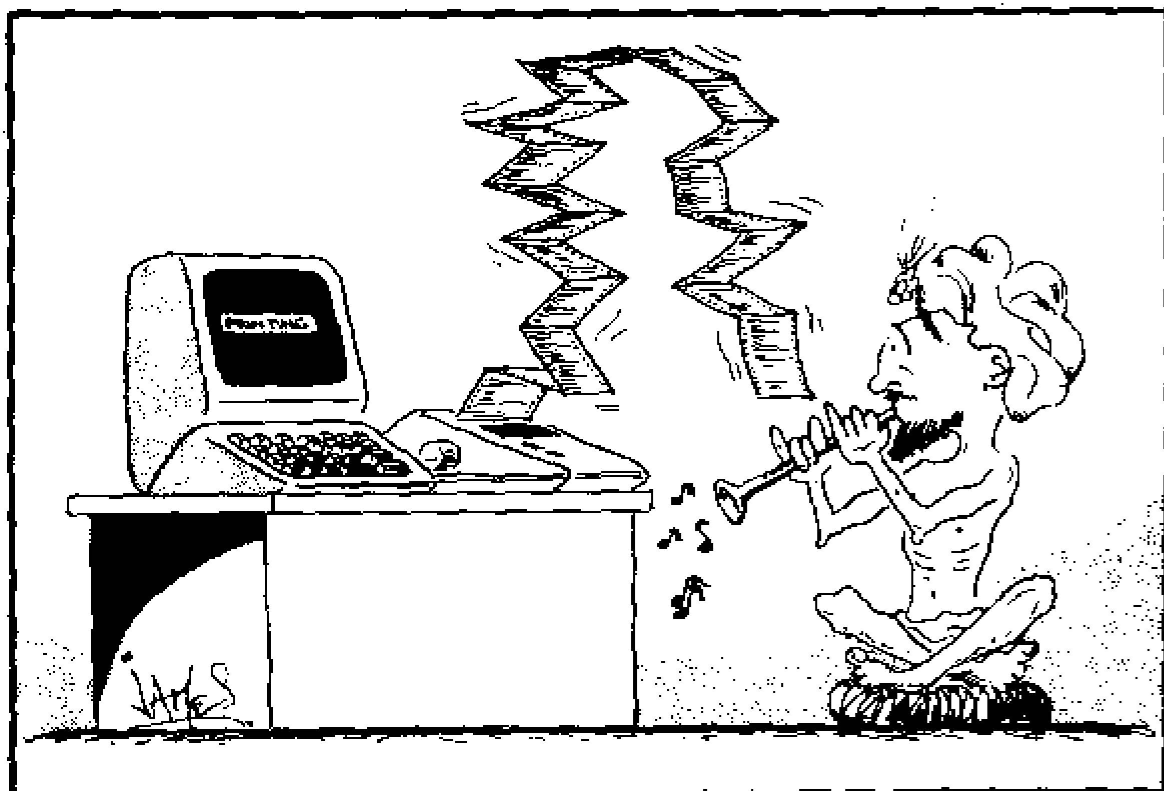
Daar het programma tijdens het backup proces de Read-character-routine verplaatst (dit om een keyboard-scan na RECORD TAPE te onderdrukken) mag het programma niet gestopt worden met <escape> tijdens het van schijf halen van een programma.

Geheugengebruik:

Het programma gebruikt als buffergebied het geheugen van #2900 tot #8000. Verder het geheugen van #8200 tot #A000. Hier staat het programma zelf, het gebied van TOP tot #A000 wordt gebruikt voor de verschillende arrays en de assembler routines.
Verder nog het zero-page gebied van #70 - #7A.

80-kolommen:

Het zal een ieder duidelijk zijn dat op een 80-kolommen scherm een veel beter overzicht mogelijk is van de inhoud van een bepaalde directory. Van dit programma bestaat dan ookal een voorlopige 80-kolommen versie; deze werkt op het moment echter alleen op een Elektuur 80-kolommen kaart welke omgebouwd is voor High- en Low-video (waarover U elders in dit Broodje meer kunt lezen, als ales goed gaat tenminste). Wanneer het programma zodanig 'verbouwd' is dat het ook op een normale 80-kolommen kaart draait, dan zal dit ook verspreid worden binnen de club. Dit programma kan dan echter wel meer dan de nu geplubiceerde versie; hetgeen betekent dat die niet meer op het normale scherm za6 draaien. De liefhebbers moeten dus nog maar even geduld hebben.



De in Acorntjesbrood 4.3 afgedrukte listing van Disc-disc copy was niet geheel in overeenstemming met de verspreide versie.De oorzaak hiervan was dat terwijl het Broodje al bij de drukker was er nog een foutje in het programma boven water kwam. Bij het copieren van een geheel volle schijf werden de laatste twee tracks niet mee gecopieerd.
Voor alle duidelijkheid volgt daarom hier nogmaals een listing van het goede programma (datum 860827).

Opmerking:

Het programma gebruikt nu tijdens het copieren als bufferruimte het geheugengebied van #2000 tot #8000.

```
  10 PROGRAM DISC-DISC COPY
  20
  30 REM GDOS 1.5
  40 REM DOUBLE DENSITY ONLY!!
  50 REM DATE: 860827
  60
  70 PROC WISH
  80    F=0;Z=0;J=1;H=1
  90    PRINT$12"WHICH DRIVE CONTAINS THE SOURCE DISC [0/1/2/3] ?"
 100    SIDE;S=K-48;@=2;P.S'
 110    PRINT"- NUMBER OF TRACKS [40/80] ?"
 120    TRACKS;V=(K-48)*10;@=3;P.V''
 130    PRINT"WHICH DRIVE CONTAINS THE TARGET DISC [0/1/2/3] ?"
 140    SIDE;O=K-48;@=2;P.O'
 150    PRINT"- NUMBER OF TRACKS [40/80] ?"
 160    TRACKS;W=(K-48)*10;@=3;P.W''
 170    IF (S+O<2) OR (S+O=2 AND S=1) OR (S+O=4 AND S=2) OR (S+O>4);F=1
 180 PEND
 190
 200 PROC SIDE
 210    DO
 220       INKEY K
 230    UNTIL K<52 AND K>=48
 240 PEND
 250
 260 PROC TRACKS
 270    DO
 280       INKEY K
 290    UNTIL K=52 OR K=56
```

```
300 PEND
310
320 PROC INIT
330   IF F;PRINT"INSERT TARGET DISC & PRESS KEY"';INKEY Q
340   *OSCLI "DIR",O,$13
350   IF F;PRINT"INSERT SOURCE DISC & PRESS KEY"'';INKEY Q
360   *OSCLI "DIR",S,$13
370   M=0;N=0;T=1;U=1;L=#60;P=#60;R=#60
380   C=(V*16)/L
390   E=?(#2506)*256+?#2507
400   G=E+?#2505+1
410   B=(G/L)+1
420   IF B>C;B=C+1;D=1
430   IF G>640 AND W<V;Z=1
440 PEND
450
460 WISH
470 INIT
480 IF Z OR (V<>W AND F);PRINT"UNABLE TO BACKUP !"'$7$7$7;END
490
500 @=0
510 PRINT"SECTORS TO COPY  : #"&G'
520 PRINT"- OUT OF         : #"&(V*16)''
530 IF F;PRINT"YOU MUST "B" TIMES CHANGE THE DISCS"'''
540
550 FOR I=1 TO B
560   *OSCLI "DRIVE",S,$13
570   *OSCLI "#",V,$13
580   XIF F;P.$11"INSERT SOURCE DISC & PRESS KEY"';INKEY Q
590   ELSE XIF M<>0;?#BFC1=M;?#BFC2=T
600        ELSE
610   *OSCLI "RSECT",&M,$32,&T,$32,&P,$32,&#2000,$13
620   IF J;M=M+6;*OSCLI "RSECT",&M,$32,&T,$32,&#1,$32,&#8E00,$13
630   IF (M=36 AND D=1 AND V=40);P=#40;J=0
640   IF (M=78 AND D=1 AND V=80);P=#20;J=0
650   *OSCLI "DRIVE",O,$13
660   *OSCLI "#",W,$13
670   XIF F;P.$11"INSERT TARGET DISC & PRESS KEY"';INKEY Q
680   ELSE XIF N<>0;?#BFC1=N;?#BFC2=U
690        ELSE
700   *OSCLI "WSECT",&N,$32,&U,$32,&R,$32,&#2000,$13
710   IF H;N=N+6;*OSCLI "RSECT",&N,$32,&U,$32,&#1,$32,&#8E00,$13
720   IF (N=36 AND D=1 AND W=40);R=#40;H=0
730   IF (N=78 AND D=1 AND W=80);R=#20;H=0
740 NEXT
750
760 PRINT"BACKUP PROCESS COMPLETED"''
770 *OSCLI "CAT",O,$13
780 END
```

PICTUREWARE: Video Digitizer Deel 2

In het Broodje nr4 heb je kunnen lezen hoe je met beperkte middelen een video-digitizer kunt bouwen van zeer acceptabele kwaliteit. De echte kwaliteit moest nog blijken omdat we maar met 1 grijsnivo (= intensiteit) werkten. In de graphics-wereld heet dit 'treshhold sampling', d.w.z. men stelt een nivo in (bij ons was dit 0 t/m 7) en het videobeeld werd opgebouwd volgens:

```
IF I(x,y) > T
THEN
    pixel (zwart)
ELSE
    pixel (wit)
END-IF
```

waarbij hier I(x,y) de te plotten pixelwaarde en T de treshhold waarde is.
We waren allang blij dat we dit soort plaatjes konden maken en de resultaten waren niet slecht. Maar omdat de digitizer de mogelijkheid heeft om een videosignaal te verdelen in 8 intensiteiten, schreeuwde dit om de benodigde software (dus geen extra hardwae !!!!!).
Het nadeel van de digitizer is, dat het maar een bit per sampling kan doorgeven. Om de benodigde 8 intensiteiten en dus 3 bits te verkrijgen, dienen we het video beeld 3 maal te samplen. In praktijk komt dit neer op ongeveer 0.8 seconde. Erg snel bewegende beelden gaan voor de volgende toepassingen dan ook niet op.
Over hoe en wat de software allemaal uithaalt om de 3 bits een videosignaal te samplen en op te slaan in het geheugen volgt geen uitleg, want alle benodigde scan-, display- en printroutines worden in een speciale digitizer-ROM gestopt die hopelijk volgend broodje gereleased kan worden.

Nu we de 3-bits videoinformatie in de computer hebben willen we het resultaat op het beeldscherm zien. We moeten dus zoeken naar een dsiplay methode die het plaatje gedetaileerder dan voorheen op het beeldscherm kan vertonen, m.a.w. we mogen geen resolutie verlies hebben. Hiervoor zijn verschillende methodes ontworpen, welke zeer fraaie resultaten geven. De bekendste methode is die van Floyd en Steinberg. Deze heren hadden het volgende bedacht: Begin boven in de linker-boven-hoek en distribueer fouten naar omliggende pixels. Dit wordt gedaan om de fouten welke veroorzaakt worden door treshhold sampling evenredig te verdelen. Het distribueren gaat van links-boven naar rechts-onder. Het Floyd-Steinbrg algorithme distribueerd 3/8*error naar rechts, 3/8*error naar beneden en 1/4*error diagonaal. Deze methode geeft een gedetaileerd beeld van videoinformatie in meer dan 1-bit.
Om software technische redenen is niet gebruik gemaakt van deze methode, maar van een algorithme welke hetzelfde prachtige resultaat, zonder resolutie verlies, op het beeld weergeeft. Dit is het zgn. Ordered dither.
Deze methode 'probeert' een error te genereren in het

plaatje. De waarde van de fout wordt opgeteld bij de waarde van de desbetreffende pixel voordat het vergeleken wordt met de geselecteerde treshholdwaarde. Het optellen van een willekeurige fout geeft niet het gewenste resultaat, alhoewel een patroon van fouten welke het beste resultaat geeft bestaat. Het fout-patroon wordt opgeteld bij het beeld in een schaakbord-patroon. Deze techniek wordt genoemd: Ordered Dither (het geordend chaos scheppen ?!).
De kleinste order-matrix is 2*2, maar omdat we 8 intensiteiten hebben is een matrix gewenst van 4*4.

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

Het ordered-dither algorithme:

Xmin, Xmax, Ymin, Ymax zijn de raster limieten voor elke scan va boven naar beneden (resp. 0, 256, 0, 192)
MOD is de modulo functie (bij de ATOM --> %)
n is matrix breedte

```
FOR y = Ymax TO Ymin STEP -1
  FOR x = Xmin TO Xmax

    bereken entry in dither matrix
    i = (x MOD n) +1
    j = (y MOD n) +1
    bepaal pixel display waarde
    IF I(x,y) < D(i,j) THEN
      Pixel (x,y) = wit
    ELSE
      Pixel (x,y) = zwart
    END-IF

    display pixel

  NEXT x
NEXT y

END
```

Mensen die opletten zullen direct zien dat het plaatje wel erg onevenredig wit moet worden omdat we maar 8 nivo's hebben (0 t/m 7) en 16 entries in de matrix. Om met een beperkt aantal nivo's toch de juiste entry in de matrix te krijgen, is gebruik gemaakt van een methode van de heer C.A.Calkins.

```
SCALE = (MAX-MIN)/N
INDEX = (VALUE-MIN)/SCALE
if INDEX > N-1 then INDEX = N-1
```

Hierbij is MAX de maximale waarde (7)
MIN de minimale waarde (0)
N de aantal waarden in de array (16)
VALUE intensiteits waarde
SCALE de schalingsfactor en
INDEX de entry in de matrix

Dan worden dit:

intensiteit:	0	1	2	3	4	5	6	7
matrix-entry:	0	2	4	6	9	11	13	15

Veelzeggend is altijd een voorbeeld. Het menselijk gezicht leent zich daar het best voor omdat het gezicht van ieder mens gelijk is op (voor ons goed zichtbare) details na. Ik ben dan ook begonnen met het scannen van Melina uit de James Bond-film 'For Your Eyes Only'. Het resultaat en het verschil tussen het samplen met 1 intensiteit of 8 intensiteiten is duidelijk zichtbaar.
Ideaal zou zijn het samplen met behulp van een video camera. Mensen houden het echt wel vol om 0.8 seconde stil te zitten en de resultaten zijn echt niet normaal meer.
Melina:



voor dither (treshhold, 1 nivo) na dither (8

Hierbij moet nog vermeldt worden dat de waarde 0 echt altijd als wit wordt afgedrukt omdat anders een soort schaakbord-wit ontstaat wat ook gebruikt wordt in de kranten (wit is daar niet wit!!). Zie algoritme (IF 0<0 is niet waar ---) pixel zwart). We krijgen dus om de vier pixels een pixel die zwart wordt. Dit zal in de digitizer-ROM wel ingebouwd worden als de zgn. paper-mode met daarnaast nog verschillende mogelijkheden om inteniteiten softwarematig te veranderen en zgn. "uploadings" van de dithermatrix. Hierbij wordt de matrix door elkaar gehusselt zodat men andere grijseffecten verkrijgt. Dat men niet altijd het algoritme kan loslaten op ieder videobeeld mag duidelijk zijn als men kijkt naar

het logo van een bekend biermerk. Het opkrikken van de detailering zou hier alleen maar storend werken.



Conclusie; bespaar iet op de digitizer voor wat betreft het aantal intensiteiten. De plaatjes die we hiermee kunnen vertonen en het nog te beschrijven printen in zgn. halftoning zijn deze investering dubbel en dwars waard.

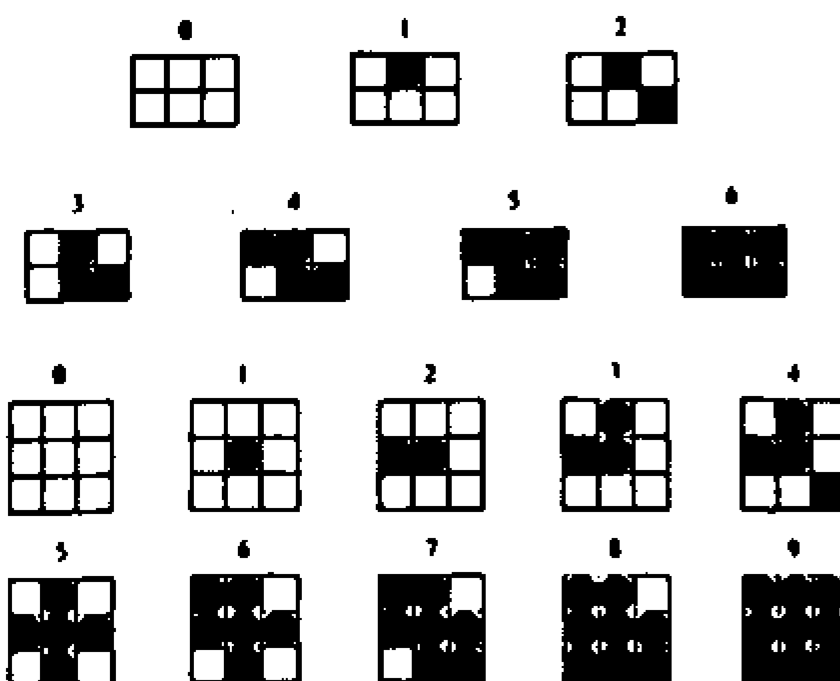
PICTUREWARE: Video Digitizer Deel ad-1

In deel 1 werd een beginnerspakket voor de digitizer beschreven. Hier bleek dat als men de contrast te ver open draaide, de software (en eigenlijk dus ook de hardware) de syncsignalen wel eens kon missen. Hierdoor ontstaan plaatjes waarvan elke kolom van 16 pixels iets hoger of lager ligt dan de voorgaande kolom. Erg storend. Hiervoor heb ik de volgende oplossing: schrap de 40-lijnen wachtlus en vervang deze door een tijd-loop.
Praktijk: verwijder regel 790 en 810 t/m 840
Voeg toe: 790 LDX @#80
810 PHA;PLA;PHA;PLA
865 LDA PORTB

Gerrit Hillebrand

PICTUREWARE: Video Digitizer Deel 3

Halftoning is een techniek die met resolutieverlies een zeer gedetailleerde tekening kan weergeven op papier. Bij het beeldscherm moesten we erop letten dat de resolutie van het beeldscherm gelijk bleef, doch als we een videobeeld willen dumpen op papier, dan is resolutie verlies (dus een grotere resolutie op papier) juist zeer wenselijk. Halftoning is een techniek die m.b.v. een zeer gering aantal intensiteit nivo's (in ons geval 8, zie Video Digitizer deel 2) een grotere visuele resolutie (en dus detail) kan weergeven. De halftoning techniek is al zeer oud en werd voor het eerst toegepast in het weven van zijde. Moderne halftoning technieken werden ontwikkeld door Stephen Hargon in 1880. Met behulp van deze techniek kan men een grote varieteit aan fotographische grijs nivo's bereiken, gebruik makend van een 2-nivo diplay medium: zwarte inkt op wit papier. Het is dan ook niet verwonderlijk dat dit ontwikkeld werd t.b.v. de drukkunst en deze technieken worden nog steeds toegepast in de drukwereld, zij het nu met veel grotere resoluties (meer dots per inch)
De techniek die hier toegepast wordt is ook bekend onder de naam 'patterning'. Men koppeld aan elke intensiteit een patroon van 2*2, 2*3 of 3*3 dots. In ons geval hebben we 8 nivo's, dus in aanmerking komend voor gebruik zijn 2*3 en 3*3.
De patronen die gebruikt worden in de drukwerk industrie zijn:



In het geval van 2*3 patronen zou men dus intensiteiten verwachten van 0 t/m 6. Maar we hebben 0 t/m 7. Volgens de regel van C.A. Calkins (Video Digitizer deel 2) krijgen we nu de volgende patronen gekoppeld aan de grijs-nivo's:

intensiteit:	0	1	2	3	4	5	6	7
patroon:	0	1	2	3	4	5	6	6

In het geval van 3*3 patronen hebben we meer patronen als

intensiteiten: 10 patronen en 8 intensiteiten. Geen probleem, de regel van Calkins biedt ons uitkomst:

intensiteit:	0	1	2	3	4	5	6	7
patroon:	0	1	2	4	5	7	8	9

Indien we nu elke intensiteit netjes omzetten in zo'n patroon, dan krijgen we de mooiste tekeningen op papier. Op dit moment kan ik dumps maken met 3*2-patronen in de printermode:640 dots/line (semi-plotter mode, welke erin geplaatst is door Ronald Boers: mijn dank hiervoor) en 3*2-patronen in de printermode:576 dots/line (plotter mode) en 3*3 patronen in de printer mode:960 dots/line. Allen EPSON-compatibele escape-protocollen, maar dat is vanzelfsprekend.

Als voorbeeld zien we hiernaast het voorbeeld van Melina (deel 2) in de 3*2/semi-plotter mode en daarna Melina in de 3*3 mode.
Let wel: de intensiteiten op papier bedragen nu: 768*855 voor 2*3 semi-plottermode en 768*768 voor 3*3 patronen.

Al deze mogelijkheden zijn vanzelfsprekend in de digitizer-ROM gebouwd, waarvan we in het volgende broodje de beschrijving mogen verwachten. U ziet, de super-eenvoudige digitizer is nu wel op z'n best gebruikt en maakt alle andere graphicdumps maar zeer povertjes.

Literatuur:
Procedural Elements for Computer Graphics - Rogers

Gerrit Hillebrand

AECOM (Atom-Electron COMmunicatie)

Een aantal leden van de club is in bezit van twee verschillende computers van het merk ACORN. Meestal zijn dit een Atom en een Electron maar ook wel een Atom en een BBC.
Vaak wil men files van het ene systeem oversturen naar het andere b.v. om gebruik te maken van een diskdrive of een printer die op het andere systeem aangesloten is.(in ons geval de Atom).
Dit is door de verschillende cassetteformaten echter niet zondermeer mogelijk. Het programma AECOM maakt een uitbreiding op het bestaande COS, waardoor files van het Electron/BBC formaat ingelezen of verzonden kunnen worden. Om het programma te kunnen gebruiken moeten Atom en Electron door een zgn. kruiskabel met elkaar verbonden worden.Dat wil zeggen de cassette ingang van de Atom ligt aan de uitgang van de Electron en omgekeerd. Na het runnen van het programma op de Atom zijn op de Atom twee nieuwe commando's beschikbaar n.l. *ELOAD en *ESAVE.
*ELOAD moet gevolgd worden door een adres. Na RETURN zal de Atom de file die door de Electron verstuurd wordt vanaf dat adres in zijn geheugen opslaan. *ESAVE moet als volgt gebruikt worden; *ESAVE "FILENAME" SSSS LLLL EEEE. FILENAME is de naam van de file die verstuurd moet worden. SSSS is het startadres in het geheugen van de Atom. LLLL is het eindadres PLUS EEN !!! in de Atom. EEEE is het executieadres en mag eventueel weggelaten worden. Met deze routines zijn heel leuke dingen te doen. B.v.het uitprinten op de Atom van een programma in BBC-basic.
Tik in de Electron in direct mode in;

```
*KEY1*SPOOL"print"|M|MLIST|M*SPOOL|M
```

en in de Atom;

```
 10 REM PRINT
 20 DO
 30 *ELOAD 3000
 40 P=#3000
 50 DO;P=P+1;UNTIL ?P=13;P=P+1
 60 DO
 70 PRINT $2
 80 PRINT $P$13;P=P+LEN(P)+1
 90 UNTIL !P=#50532A3E
100 PRINT $3
110 UNTIL 0
```

Een file in de Electron kan nu op de printer van de Atom worden afgedrukt door in de Electron op F1 te drukken (nadat bovenstaand programma op de Atom is gerund).Op dezelfde manier kunnen files van de drive naar de Electron gestuurd worden.

Marien van Westen

#BXXX

Het zal u bekend zijn dat de standaardisatie van van alles en nog wat een wereldprobleem is. Wij lopen met onze Atom dan ook niet geheel achteraan, want om te beginnen heeft ACORN voor de Atom wat grenzen afgebakend en daarna is er toch zoiets als een "standaard Atom" in de club gedefinieerd. Het zal u natuurlijk niet verbazen dat zulks vooral bepaald is door de geschiedenis. Dat groeiproces is echter nog geen garantie voor een optimaal resultaat. Ikzelf vind het I/O gebeuren enigzins ongelukkig.Om te beginnen is daar meneer ACORN die het #Bxxx-gebied voor I/O bestemde, maar wel meteen zondigde door zijn FDC (floppy disc controller) op #A00/#A01 te adresseren. Misschien dat hij vond dat er met 1K voor de VIA en 1K voor de PIO wat weinig ruimre overbleef in het B-blok. Welnu,op deze manier vinden wij dat allemaal wel ,denk ik. Er kwam toen iemand met het geniale idee om de FDC op het E-blok te adresseren. Dat idee werd snel door de club afgekraakt met het enige argument dat zw daarvoor konden verzinnen : er bestaat programmatuur die geen vrije bytes toelaat op het E-blok! Je zou misschien verwachten dat er dan een of andere normstelling zou ontstaan om de ruimte in het B-blok te verdelen, maar nee hoor, er kwam een 80-kolomskaart die geadresseerd was op #BE2x. Hoe verzinnen ze het?
Ik zal in het kort mijn idee uiteen proberen te zetten.
Een kleine voorziening geeft de VIA de 16 adressen die hij nodig heeft, zodat de rest van #B800 - #BBFF vrij blijft (voor prive I/O). Diezelfde voorziening kent ook 16 adressen toe aan de 8255. Dat zijn er wel 12 teveel, maar de rest van #B000 - B3FF blijft weer vrij (voor club I/O). Zo kan #B400 - #B7FF vooralsnog in reserve gehouden worden, dat is leuk voor de eenvoudige knutselaar. Aangezien #BFFF als schakelbyte in wezen als RAM gebruikt wordt en er genoeg overtollige 2114's zijn die er gemakkelijk bijgeplaats kunnen worden, zou #BC00 - #BFFF Ram moeten worden voor onregelmatige opslag zoals voor de Floating Point. P-CHARME, PFC enz. Dat idee is al eens gepubliceerd. De eenvoudige knutselaars wachten nu dus met smart op de aanwijzigingen van de club om de adressering van hun 80-kolomskaart zonder beschadigingen om te kunnen bouwen. Hoe de Atom aangepast moet worden zal ik hierna proberen uit te leggen. Voor een investering van pakweg EEN HELE GULDEN ontstaat er een zee van ruimte voor u op #B0xx en #B8xx. Voor dir bedrag schaft u zich een HCF4078 aan en wanneer u dat niet in voorraad heeft, twee weerstanden van 1K5 en twee dioden. De 4078 is een OR/NOR poort. De adreslijnen A4 - A9 staan op de ingangen. Met de OR-uitgang worden de 6522 en de 8255 uitgeschakeld. Fig.1 toont de omgeving van IC49 aan de onderzijde van de print (toetsenbordzijde). Met een scherp mesje kerft u dan een onderbreking in de koperbanenop de twee plaatsen die door een pijl zijn aangegeven.(De oude situatie kan eenvoudig hersteld worden door een druppel tin over de kerf aan te brengen.) De gemaakte onderbreking overbrugt u dan met een weerstand, zoals in fig.2 is aangegeven. De adressering

zal dan gewoon blijven werken. Wanneer echter de chip-selectlijn kunstmatig hoog wordt gehouden,(dat mag vanwege die weerstand),dan denkt de VIA/PIO dat hij niet wordt aangeroepen. Deze truc wordt verzorgt door de 4078.Dit IC wordt ondersteboven gesoldeerd aan een aantal pootjes van IC-voet 10. In fig.3 ziet u de bovenzijde van de 4078.De dik getekende pootjes zijn omgebogen,zodat ze naar boven wijzen.Aan die pootjes moet hij worden vastgesoldeerd. In fig.4 ligt hij daartoe op zijn rug. Ik heb hem iets te klein getekend, zodat duidelijker te zien is wat waaraan gesoldeerd is. Vergeet het draadje aan de min-poot niet en plaats de twee diodes. Een hexdump vanaf #B000 moet tonen of het werkt.Het lijkt mij dat dit voor de echte minima (zij die nog nooit iets met hun Atom gedaan hebben), geen onoverkomelijke investering hoeft te zijn. Er zijn natuurlijk ook namaak minima (zij die ooit wel eens geprobeert hebben om een of ander IC in een of andere voet te steken). Ook voor hen hoeft dit geen wezenlijk probleem op te leveren. Het wordt dan misschien weer eens tijd voor de club om opnieuw een MODALE Atom te definieren. Het lijkt de grote-mensenwereld wel!

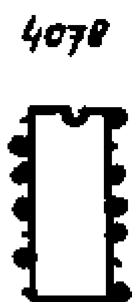


FIG.3

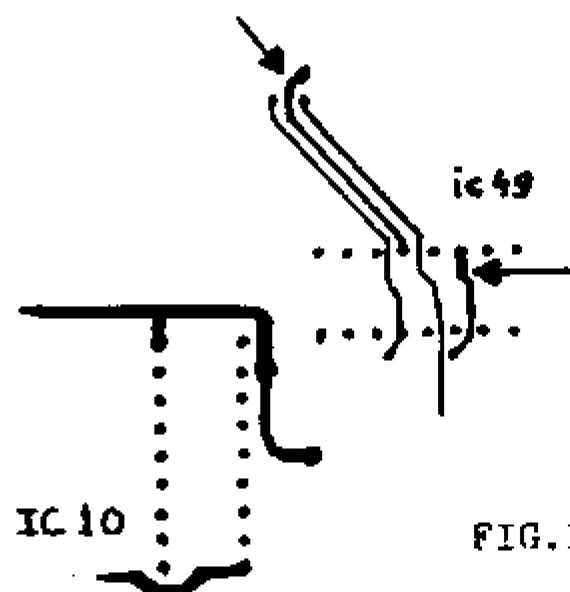


FIG.1

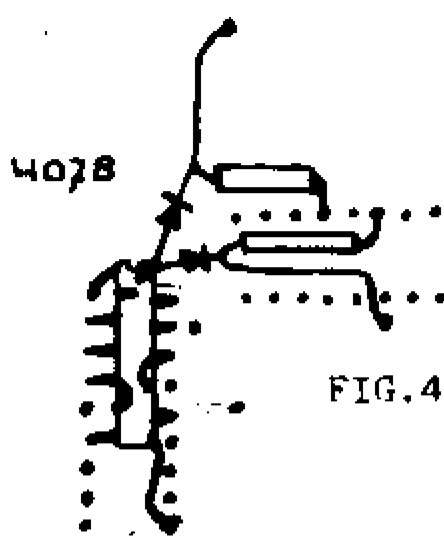


FIG.4

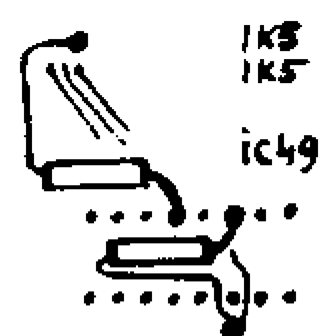


FIG.2

Wim Osterholt

Wilt u lid worden van de ATOM COMPUTER CLUB?
Neem dan contact op met de penningmeester van de regio waar u bij ingedeeld wilt worden.Deze kan u inlichten omtrent het lidmaatschap.

Regio NOORD;
D.Uuldriks Wiemers 14 9642 KG Veendam
05987-19611

Regio OVERIJSSEL/GELDERLAND
N.van Wijnen Korte Dreef 12 8302 BS Emmeloord
05270-13583

Regio TWENTE;
W.Buning Kieskamp 1 7783 EB Gramsbergen
05246-1831

Regio NOORD-HOLLAND;
P.van Kuik Zuideinde 54-a 1843 JP Groot-Schermer
02997-1902

Regio DEN HAAG;
Th.WaayerL.Couperusstraat 6 2274 XP Voorburg 070-862504

Regio DELFT;
F.von Morgen M.de Ruyterweg 21 2628 BA Delft

Regio ROTTERDAM;
R.de HaanBrasem 125 2986 HA Ridderkerk 01804-25160

Regio CENTRUM;
P.van Mourik Ruiterstede 60 3431 XN Nieuwegein
03402-48781

Regio ARNHEM;
J.Hartog Keyenbergseweg 60 6871 WK Renkum 08373-13757

Regio ZEELAND;
E.GijsselDorpsstraat 86 4424 CZ Wemelding 01192-2121

Regio BRABANT-OOST;
P.Ehrlig Roostenlaan 266 5644 BS Eindhoven 040-114183

Regio LIMBURG;
A.van Zantvoort Mozartstraat 328 6044 RS Roermond
04750-21797

Regio BELGIE;
R.Leyssens Oude Baan 127 3550 Heusden Belgie

Bij het aangaan van het lidmaatschap kunt u de contributie overmaken op de rekening van de federatie.Vermeld hierbij uw volledige naam, adres en de regio waar u bij ingedeeld wilt worden.